Timo Denk

Alte und neue Zöpfe

Neuerungen und Inkompatibilitäten in PHP 7.0

Ende dieses Jahres soll das nächste große Release der Skriptsprache PHP erscheinen. PHP 7.0 enthält viele Verbesserungen und neue Features – aber auch einige Rückschritte und Inkompatibilitäten zu älteren PHP-Versionen.

Am 12. November 2015 soll die PHP-Version 7.0 das aktuelle PHP 5.6 ablösen, im Moment liegt der dritte Release Candidate von PHP 7 vor. Die Version 6 lässt das PHP-Team einfach aus: Die Entwicklung daran ist 2010 eingeschlafen und die meisten dafür angekündigten Features sind bereits in die Versionen 5.3 und 5.4 eingeflossen.

Die Entwickler preisen vor allem die Performance der neuen PHP-Version: „up to twice as fast as PHP 5.6“ heißt es auf php.net. In unseren Experimenten mit einem gut gefüllten Wordpress-Server beantwortete PHP 7.0 RC3 tatsächlich doppelt so viele Anfragen pro Sekunde wie die noch aktuelle Version 5.6.12. Bei Unit-Tests schnitt PHP 7 noch besser ab und konnte beispielsweise Mathematik-Funktionen über dreimal so schnell abarbeiten wie der Vorgänger. Webhoster dürften sich über den enormen Performancezuwachs am meisten freuen, da mit dem Upgrade zu PHP 7 ein Server von einer größeren Kundenanzahl verwendet werden kann. Das spart Anschaffungs- und Betriebskosten.

Ein weiterer Grund, sich prontissimo mit PHP 7 zu beschäftigen: Der Support mit Sicherheits-Updates für Version 5.4 läuft im September 2015 aus, für 5.5 im Juli 2016.

Konsistenter

Darüber hinaus wartet PHP 7 mit zahlreichen sprachlichen Neuerungen auf. Zum Beispiel wurden Inkonsistenzen in der Auswertungsrichtung von Ausdrücken beseitigt. So wird etwa bei der Verwendung von variablen Variablen in Verbindung mit Arrays die Reihenfolge der Auswertung umgekehrt: Diese läuft nun wie bei anderen Ausdrücken von links nach rechts. Die nebenstehende Tabelle zeigt, welche Bedeutung das auf Ausdrücke hat und wie die alte Funktion unter Verwendung von geschweiften Klammern wiederhergestellt werden kann.

Achtung: Die neue Auswertungsrichtung birgt Inkompatibilitäten zu älteren PHP-Versionen. Wenn Sie in Ihrem Code variable Variablen in Verbindung mit Arrays verwenden, sollten Sie prüfen, ob das gewünschte Verhalten unter PHP 7 Bestand hat.

Als `global` gekennzeichnete variable Variablen dürfen gar nicht mehr ohne die Klammerung geschrieben werden. Die Zeile

```
global $$foo->bar;
```

muss unter PHP 7 wie folgt lauten:

```
global ${$foo->bar};
```

Auch das `list()`-Konstrukt arbeitet nun bei der Zuweisung von Werten nicht mehr rückwärts. Nach der Ausführung des folgenden Codes hätte bei alten PHP-Versionen die `Array`-Variable `$array[]` den Wert `[3, 2, 1]` gehabt:

```
list($array[], $array[], $array[]) = [1, 2, 3];
```

Mit PHP 7 ist das Ergebnis logischer: `[1, 2, 3]`. Das entspricht der Reihenfolge bei der Zuweisung an Nicht-Arrays:

```
list($a, $b, $c) = [1, 2, 3];
echo $a, $b, $c; // ergibt 123
```

Darüber hinaus sind leere `list()`-Zuweisungen nicht mehr erlaubt. Die folgenden drei Code-Zeilen bemängelt PHP 7 mit „Fatal error: Cannot use empty list“:

```
list() = $a;
list(,,) = $a;
list($x, list(), $y) = $a;
```

Typsicherer

Zu konsistentem Code trägt auch Typsicherheit bei. Die gewährleistet PHP auf Wunsch bei Funktionsparametern und -rückgabewerten. Musste man Typen bislang mit `is_numeric()` oder `is_bool()` prüfen, geht das jetzt viel einfacher mit Typangaben. Sie stehen bei Funktionsparametern jeweils vor dem Argument, bei Rückgabe hinter der Parameterliste mit einem vorangestellten Doppelpunkt:

```
function add(int $a, int $b): int {
  return $a + $b;
}
var_dump(add(1, 2)); // int(3)
```

Aber Achtung: Die automatische Typkonvertierung bewirkt bei einem Funktionsaufruf mit zwei in Zahlen konvertierbaren Strings, beispielsweise `"1"` und `"2"`, dass PHP die Strings in die Zahlen `1` und `2` umwandelt. Es bleibt daher beim ganzzahligen Rückgabewert `3`.

Um die automatische Konvertierung der Übergabeparameter zu verhindern, schreiben Sie am Anfang eines PHP-Skripts

Neue Auswertungsrichtung

Ausdruck	alte Bedeutung	neue Bedeutung
$$foo['bar']['baz']	${$foo['bar']['baz']}	($$foo)['bar']['baz']
$foo->$bar['baz']	$foo->{$bar['baz']}	($foo->$bar)['baz']
$foo->$bar['baz']()	$foo->{$bar['baz']}()	($foo->$bar)['baz']()
Foo::$bar['baz']()	Foo::{$bar['baz']}()	(Foo::$bar)['baz']()

```
declare(strict_types=1);
```

Funktionsaufrufe mit falschen Datentypen führen dann in jedem Fall zum Abbruch mit obiger Fehlermeldung.

Bei Aufrufen mit nicht konvertierbaren String-Werten, zum Beispiel "c" und "t", bricht das Programm die Ausführung mit einem „Fatal error: Uncaught TypeError“ ab.

Die neue Typprüfung in PHP hat ihren Preis: Die Ergebnisse unserer Benchmarktests zeigen, dass die Festlegung von Parameter- und Rückgabetypen die Ausführungszeit der add()-Funktion um rund 50 Prozent verlängert. Werden nur die Parametertypen angegeben, wächst sie um rund 30 Prozent. Bei ausschließlicher Angabe des Rückgabetyps dauert die Ausführung circa 13 Prozent länger.

Die Typprüfung scheint mit unverhältnismäßig hohen Performance-Einbußen einherzugehen. Zum Glück trügt der Schein, denn der Funktionsrumpf von add() ist ja nur eine Zeile lang. Bei umfangreicheren Funktionen macht die Typüberprüfung daher einen erheblich geringeren Teil der Ausführungszeit aus.

Den obigen Fehler konnte man in früheren PHP-Versionen nur abfangen, indem man ihn in der catch-Klausel explizit angab:

```
try {
  echo add("foo", "bar");
}
catch (TypeError $e) {
  // Fehlerbehandlung ...
}
```

Als allgemeine Exception ließ er sich aber nicht abhandeln, denn ein TypeError war nicht vom Typ Exception. Mit PHP 7 leiten nun alle Fehler und Exceptions von Throwable ab. Deshalb lassen sich nun alle mit einer einzigen catch-Klausel abfertigen:

```
try {
  echo add("foo", "bar");
}
catch (Throwable $e) {
  // Fehlerbehandlung ...
}
```

Neue Operatoren

PHP ist nicht nur schneller, konsistenter und typsicherer geworden, sondern auch kompakter, zum Beispiel bei bedingten Zuweisungen der Art

```
$c = ($a === NULL) ? $b : $a;
```

Sie sieht man so häufig in Quelltexten, dass PHP 7 dafür eine Abkürzung mitbringt, den sogenannten Null-Coalescing Operator ??, mit dem man Obiges wie folgt schreiben kann:

```
$c = $a ?? $b;
```

Der ??-Operator gibt also den links stehenden Wert zurück, wenn dieser nicht NULL ist, andernfalls den rechten Wert.

Auch längere Verkettungen à la $a ?? $b ?? $c ?? "alles null" sind möglich und werden von links nach rechts abgearbeitet.

Der neue „Spaceship Operator“ <=>, genauer: Combined Comparison Operator, ist hilfreich bei Sortierfunktionen für Arrays.

Combined Comparison Operator

Operator	<=> Äquivalent
$a < $b	($a <=> $b) === -1
$a <= $b	($a <=> $b) === -1 \|\| ($a <=> $b) === 0
$a == $b	($a <=> $b) === 0
$a != $b	($a <=> $b) !== 0
$a >= $b	($a <=> $b) === 1 \|\| ($a <=> $b) === 0
$a > $b	($a <=> $b) === 1

Herkömmlicher Code in einem Komparator sieht wie folgt aus:

```
return ($a < $b) ? -1 : (($a > $b) ? 1 : 0);
```

Mit dem neuen Operator wird der gleiche Code viel kompakter:

```
return $a <=> $b;
```

Er vergleicht also zwei Werte miteinander und gibt -1 zurück, wenn der linke Wert kleiner ist als der rechte, 0 bei Gleichheit und 1, falls der rechte Wert kleiner ist als der linke.

Zusammenstreichen

In die Rubrik „kompakterer Code“ fällt auch die als „Group Use Declarations“ bezeichnete neue Funktion des Schlüsselworts use. Damit lassen sich nun mehrere use-Deklarationen vereinen. Die zwei Zeilen

```
use namespace\ClassA;
use namespace\ClassB as B;
```

werden in PHP 7 auf Wunsch zu

```
use namespace\{ClassA, ClassB as B};
```

Das funktioniert auch bei Funktionen und Konstanten:

```
use function foo\math\{ sin, cos, cosh };
use const foo\math\{ PI, E, GAMMA, GOLDEN_RATIO };
```

Ein praktisches Sprachfeature wird in Zukunft leider verschwinden: Beim Zugriff auf Funktionsparameter mit func_get_arg() gibt PHP künftig nicht mehr den ursprünglichen Wert des Parameters zurück. Wenn die übergebene Variable innerhalb der Funktion geändert wurde, liefert func_get_arg() den neuen Wert. Folgender Code-Schnipsel verdeutlicht die nicht rückwärtskompatible Änderung:

```
function foo($x) {
  $x++;
  var_dump(func_get_arg(0));
}
foo(1);
```

Ältere PHP-Versionen geben int(1) aus, ab PHP 7 hingegen int(2).

Das sind nur die wichtigsten Änderungen in PHP 7. Eine detaillierte Liste aller Neuerungen finden Sie im PHP-Repository bei GitHub [1].

Wenn für PHP 5.x geschriebene Projekte unter PHP 7 laufen sollen, steht man als Entwickler vor der anstrengenden Aufgabe, im Code nach Inkompatibilitäten zu suchen und

PHP 7 ausprobieren

PHP 7 lag zum Zeitpunkt der Drucklegung dieses Artikels als Release Candidate 3 vor. Diese Vorabversion sollten Sie nur zu Testzwecken verwenden, nicht für den produktiven Einsatz.

Wegen der möglichen Inkompatibilitäten zu bestehendem Code empfiehlt es sich, Software so früh wie möglich mit dieser Vorabversion testen. Die folgende, knappe Anleitung installiert PHP 7.0.0 RC3 auf einem Linux-Rechner.

Die erforderlichen Kommandos müssen Sie nicht von Hand abtippen, sondern können Sie in Textform über den c’t-Link am Ende des Artikels herunterladen. Nicht abgedruckt sind die Inhalte einiger Konfigurationsdateien, die Sie ebenfalls über den c’t-Link erhalten.

Alle folgenden Schritte führen Sie als root aus. Für die Installation von PHP 7.0.0RC3 auf Debian 8 (Jessie) beginnen Sie mit dem Download des Quellcodes:

```
mkdir -p /opt/php-7.0.0RC3
mkdir /usr/local/src/php7-build
cd /usr/local/src/php7-build
wget https://downloads.php.net/~ab/php-7.0.0RC3.tar.bz2
tar -xjvf php-7.0.0RC3.tar.bz2
cd php-7.0.0RC3
```

Nach dem Entpacken installieren Sie die für die Kompilierung von PHP erforderlichen Packages und starten die Installation:

```
apt-get install build-essential libfcgi-dev libmcrypt-dev libssl-dev ⤷
      libc-client2007e-dev libbz2-dev libcurl4-openssl-dev libjpeg-dev libpng12-dev ⤷
                  libfreetype6-dev libkrb5-dev libpq-dev libxml2-dev libxslt1-dev
ln -s /usr/lib/libc-client.a /usr/lib/x86_64-linux-gnu/libc-client.a
./configure --prefix=/opt/php-7.0.0 --with-pdo-pgsql --with-zlib-dir ⤷
        --with-freetype-dir --enable-mbstring --with-libxml-dir=/usr --enable-soap ⤷
    --enable-calendar --with-curl --with-mcrypt --with-zlib --with-gd --with-pgsql ⤷
 --disable-rpath --enable-inline-optimization --with-bz2 --with-zlib --enable-sockets ⤷
--enable-sysvsem --enable-sysvshm --enable-pcntl --enable-mbregex --enable-exif ⤷
       --enable-bcmath --with-mhash --enable-zip --with-pcre-regex --with-mysqli ⤷
         --with-pdo-mysql --with-mysqli --with-jpeg-dir=/usr --with-png-dir=/usr ⤷
                  --enable-gd-native-ttf --with-openssl --with-fpm-user=www-data ⤷
    --with-fpm-group=www-data --with-libdir=/lib/x86_64-linux-gnu --enable-ftp ⤷
       --with-imap --with-imap-ssl --with-kerberos --with-gettext --with-xmlrpc ⤷
                                  --with-xsl --enable-opcache --enable-fpm
make
make check
make install
```

Kopieren Sie die über den c’t-Link erhältliche Datei php.ini nach /opt/php-7.0.0/lib/php.ini.

Der PHP FastCGI Process Manager (FPM) klinkt PHP in den Webserver ein. Kopieren Sie Vorlagen für die FPM-Konfigurationsdateien in das PHP-7-Verzeichnis:

```
cp /opt/php-7.0.0/etc/php-fpm.conf.default /opt/php-7.0.0/etc/php-fpm.conf
cp /opt/php-7.0.0/etc/php-fpm.d/www.conf.default ⤷
                                        /opt/php-7.0.0/etc/php-fpm.d/www.conf
```

Nun entfernen Sie die Auskommentierung der Zeile pid = run/php-fpm.pid in der Datei /opt/php-7.0.0/etc/php-fpm.conf.

PHP-FPM empfängt an einem lokalen Port die vom Server übergebenen Daten. Ändern Sie in der Konfigurationsdatei /opt/php-7.0.0/etc/php-fpm.d/www.conf den Parameter listen auf listen = 127.0.0.1:9007.

Erstellen Sie das Init-Skript (siehe c’t-Link) für PHP-FPM und fügen Sie es zum Autostart hinzu:

```
nano /etc/init.d/php-7.0.0-fpm
chmod 755 /etc/init.d/php-7.0.0-fpm
update-rc.d php-7.0.0-fpm enable
```

Nachdem Sie die Datei /lib/systemd/system/php-7.0.0-fpm.service angelegt haben, können Sie PHP-FPM wie folgt starten:

```
/etc/init.d/php-7.0.0-fpm start
```

Fehlt noch der Webserver. Wir haben uns für nginx entschieden, weil er leicht zu konfigurieren und schlank ist:

```
apt-get install nginx
```

Kopieren Sie die nginx-Konfigurationsdatei nach /etc/nginx/sites-available/default und starten Sie nginx neu, um die Änderungen zu übernehmen.

```
/etc/init.d/nginx restart
```

Voilà! Nun können Sie PHP 7 verwenden. Dateien im Verzeichnis /var/www/html stehen lokal über die URL http://localhost/ bereit.

PHP 7 versus PHP 5

Test	PHP 7.0.0RC3 ◀ besser	PHP 5.6.12 Laufzeiten in ms[1]	Verbesserungsfaktor
Math	201	683	3,4
String-Manipulation	271	770	2,8
Schleifen	166	486	2,9
Verzweigungen (if/else)	120	295	2,5

[1] Tests siehe www.php-benchmark-script.com

Typprüfung

	strict_types=0 ◀ besser	strict_types=1 Laufzeiten in ms[1]	Performancekosten
Parameter und Rückgabewert	32,8	33	48,4 %
Parameter	29	29,6	31,2 %
Rückgabewert	25	25,8	13,1 %
keine	22,1	22,2	–

[1] 1 Mio. Aufrufe der add()-Funktion

diese zu beheben. Eine Überprüfung der Syntax übernimmt PHP mit dem Konsolenbefehl `php -l index.php`. Das reicht für einen vollständigen Kompatibilitätscheck allerdings bei Weitem noch nicht aus, da viele Änderungen keinen Syntaxfehler verursachen, sondern lediglich die Funktionsweise des Programms verändern. Fundierte Hilfe finden Sie unter [2].

Migration

Migrations-Assistenten befinden sich derzeit noch im Entwicklungsstadium. Der PHP 7 Migration Assistant Report (MAR) hat sich in unseren Experimenten als zuverlässig erwiesen. Er listet Inkompatibilitäten von wahlweise einzelnen Dateien oder ganzen Verzeichnissen in einer separaten Datei auf (Download via c't-Link). Dabei wurden nahezu alle potenziellen Problemquellen erkannt.

Epilog

PHP gelingt mit Version 7.0 ein großer Schritt nach vorne: Neue Funktionen und mehr Konsistenz sind nur einige Vorteile, die PHP verbessern und es für neue Projekte sehr attraktiv machen. Ob sich die Migration von altem Code lohnt, hängt stark davon ab, wie sehr dieser Gebrauch von inkompatiblen Funktionen macht. Die Chancen, ohne Änderungen am Code direkt mit PHP 7 starten zu können sind jedoch hoch: Der PHP 7 Migration Assistant Report hat drei der fünf am meisten beachteten PHP-Projekte bei GitHub als vollständig PHP-7-kompatibel eingeschätzt [3].

Und die PHP-Entwicklung schreitet weiter voran. Die Entwürfe im PHP-Wiki bieten bereits einen Einblick in weitere Features wie beispielsweise Konstruktoren für statische Klassen, Enumerationen, `Loop-Else`-Konstrukte und Datentyp-Angaben für Attribute (`public int $value`). Diese werden aber, wenn überhaupt, erst in Versionen jenseits der 7.0 eingebaut. (ola@ct.de)

Literatur

[1] Liste aller Änderungen in PHP 7: https://github.com/php/php-src/blob/24fc74d412/UPGRADING

[2] Migrating from PHP 5.6.x to PHP 7.0.x: http://php.net/manual/en/migration70.php

[3] Top Trending PHP Repositories: https://github.com/trending?l=PHP

c't PHP7, Skripte, Dokumentation: ct.de/yyvw