A PHP története

Hihetetlennek tűnik, de a PHP már közel tízéves múltra tekinthet vissza.

A 1994-ben ***Rasmus Lerdorf*** a saját honlapján azt kívánta megtudni, hogy kik olvassák nyilvánossá tett önéletrajzát. Erre a feladatra írt Perl nyelven egy egyszerű alkalmazást. Mivel a futtató kiszolgáló eléggé túlhajszolt volt, és a program emiatt többször feladta a harcot, Rasmus úgy döntött, hogy az egészet újraírja C-ben.
Történetesen ez a kiszolgáló, ahol ügyködött, számos más felhasználónak is otthont adott, akik felfigyeltek a munkájára. Többen megkérték, engedje meg számukra, hogy ezt a megoldást a saját lapjukon is használhassák. Rasmus hajlott a dologra, aminek rövidesen az lett a következménye, hogy egyre több kívánságot teljesíthetett programja továbbfejlesztésével kapcsolatban. Látva a dolog sikerét, munkáját leírással ellátott programcsomaggá állította össze, és levelezőlistát indított.
Ekkor kapta meg a program a nevét: Personal Home Page Tools. Ez rövidesen Personal Home Page Construction Kit névre módosult. Közben adatbázis-kiszolgálókkal is játszadozni kezdett, és összeütött egy másik alkalmazást, ami által képes volt SQL-lekéréseket összekapcsolni a hozzájuk tartozó webes űrlapokkal és listákkal. Ezt a csomagját Form Interpreter néven ismerhette meg a nagyközönség.
A dolgok felgyorsultak, 1997 végére a két program PHP/FI 2.0 néven egyesült, amihez mind a két összetevőt alaposan át kellett írni. Innentől számítható a PHP önálló programnyelvnek, olyannak, amelyet weblapokba ágyazhatóan lehetett futtatni. Mire a 2.0-s változat próbaváltozatainak a végére ért, és végleges pompájában kiadásra került, megnőtt a támogatott adatbázis-kiszolgálók száma is. E próbaváltozatok folyamán került be a MySQL-adatkapcsolat támogatása is a nyelvbe.
A PHP C nyelvű forrása ekkor még kényelmesen elfért egyetlen könyvtárban. A 2.0 végleges kiadásakor is csak a regexp-támogatás kapott kitüntetett helyet, azaz saját alkönyvtárat.

A csapatmunka eredménye

És eljött 1998 nyara, vele együtt pedig a PHP 3.0-s változata. Ez a kiadás már jelentős csapatmunka eredménye – ***Zeev Suraski*** és ***Andi Gutmans*** nagyfokú közreműködésével. Ők ketten teljesen a nulláról indulva újra felépítették a PHP parancsfájl-feldolgozó motort. Ez a mag az, amit ma Zend néven ismerünk. A PHP rövidítés ekkor nyerte el azt az értelmezését, ami jelenleg is használatos: „PHP: Hypertext Preprocessor”.
A hármas sorozat szűk kétéves fennállása során rengeteg fejlesztésen ment keresztül, de 2000 tavaszára ismét újabb nagy ugrás tanúi lehettünk.

Minőségi ugrás: a 4-es változat

A PHP 4.0 valóban ismét nagy változásokat hozott. Maga az alapnyelv is jelentősen bővült, és a rendszer magja ekkortól támogatta a különféle modulok hozzáírását. A legtöbb PHP3-ba épült támogatásból is ilyen modul készült. Egészen idáig a PHP csak az Apache webkiszolgálóval működött együtt teljes összefonódásban, míg más rendszereken csak mint CGI-alkalmazás állta meg a helyét. Egyedül a 4.0-val bevezetett egységes, webkiszolgálókkal kapcsolatot tartó alapréteg jelentett áttörést ezen

a területen. A PHP belső motorja hivatalosan ekkortól kapta meg a Zend elnevezést.

A jelen

A nagyobb változatszámugrásoknak mindig megvan az oka, miként most is. Több olyan újdonság is bekerült az új változatban, amelyek külön-külön is indokolhatták volna ezt az ugrást. Az egyik ilyen fontos lépés a CLI (parancssorbarát felület) végleges, immár nem kísérleti jellegű megjelenése. Eddig, ha bármilyen feladat ellátásához parancssorból futtatható PHP-programra volt szükség, nem létezett tökéletes megoldás. Márpedig ilyen feladat a komolyabb leterheltséggel számoló alkalmazások esetén könnyen előjöhet – elég egy komolyabb reklámcsík-kiszolgálóra gondolnunk.
A reklámok véletlenszerű elosztását ekkor összetett szempontok alapján, viszonylag bonyolultabb számításokkal határozhatjuk meg. E valószínűségeket minden egyes letöltéskor meghatározni és az alapján sorsolni túlzott és felesleges terhet jelent a gép számára. Ehelyett elegendő néhány percenként kiszámítani a megjelenési valószínűségeket, majd a következő időszakban eszerint küldeni a reklámokat, immáron letöltésenként egyetlen egyszerű kockadobással döntve. A meghatározott időközök `crontab` felállításával könnyedén létrehozhatók. Igen ám, de a PHP arra született, hogy HTTP-protokollon keresztül dolgozzon, nem pedig a héjból hívva. Arra a feladatra, hogy a PHP-kódok a `cron` segítségével mégis futtathatók legyenek, több megoldás is elterjedt:

- A Lynx szöveges böngésző felhasználásával, ami némi kivetnivalót hagy maga után, hiszen egy felesleges áttételen keresztül indítjuk be PHP-kódot.
- Az Apache-modul mellé egy CGI-változat fordításával és ennek parancssori hívogatásával. Ez már kiszolgálóbarátabb megoldás, de még mindig nem az igazi módszer.
  A 4.3.0-s változattól kezdve immár nem kell kétszer fordítanunk, ha parancssori elérést szeretnénk, és a Lynx-féle

trükközés is hamar elfelejthető. Mostantól kezdve amikor Apache-modult állítunk elő forrásból fordítással, egyben egy CLI felületű parancssorosan futtatható PHP is hadrendbe áll. Ez természetesen egy `-disable-cli` kapcsolóval kikapcsolható. Akkor sem lesz CLI felületünk, ha Apache-modul helyett a CGI-változat mellett döntünk. Lássuk, miféle kedvességekkel kényeztet minket a CLI-megvalósítás:

- Nem ad HTTP-fejléceket.
- Nem változtatja meg a pillanatnyi munkakönyvtárat a futó program könyvtárára.
- A hibaüzenetek nem kapnak HTML-formázást.
- A szabványos kimenet nem kerül átmeneti tárba, minden azonnal megjelenik a konzolon (`implicit_flush`).
- A program futási idejének nincs felső határa (`max_execution_time`).
- A meghíváskor átadott tulajdonságok a `$argc` és `$argv` változókon keresztül elérhetők. A `register_globals`-ra a CLI hasonlóképp válaszol, mintha Apache alól futna. Ennek `off` állapotba állításakor az `$argc` és `$argv` változók nem jönnek létre, viszont a `$_SERVER` tömbben továbbra is elérhetők.
- Bevezeti az `STDIN`, `STDOUT`, `STDERR` állandókat. Ezek a nevüknek megfelelő ki- és beviteli eszközökre mutatnak, használatukhoz tehát nem szükséges az `fopen()` hívása. Hasonlóképpen a bezárásukkal sem kell törődni. A CLI-t használva a PHP a Perlhez hasonlóképpen használható, azaz többféleképpen is futtathatjuk:
  1. A `php program.php` vagy a `php -f program.php` segítségével.
  2. A `-r` kapcsoló segítségével közvetlen parancsokat tudunk végrehajtatni: `php -r 'print_r(get_defined_constants());'`
  3. A bemenetére csöveken át is csatlakozhatunk.
  4. `#!/usr/bin/php` sort biggyesztve a PHP-parancsfájl elejére, azt önállóan is futtathatóvá tehetjük.

Ugyancsak most bevezetett újdonság az egységesített adatfolyamkezelés (Stream). A dolog szellemisége a Unix „minden fájl” felfogásához hasonlít. Itt arról van szó, hogy minden ki- és bemeneti művelet nagyon hasonlatos egymáshoz, legyen szó akár fájlba írásról, csővezetékből való adatfogadásról, memóriakezelésről, akár foglalaton keresztüli kapcsolattartásról. Ezeket a feladatokat rengeteg különböző, ámde mégis nagyon hasonló függvénygyűjteménnyel lehetett eddig megoldani. Foglalatokhoz példaképpen a `socket_*` függvények vannak rendszeresítve, míg FTP-re rengeteg `ftp_*` függvényt találunk. Az egységesítéshez a fájlkezelő függvények átírására, valamint jó pár `stream_*` eljárás rendszeresítésére volt szükség. A meglévő függvények egy része viszont ettől fogva már csak ezeknek az adatfolyamféle megfelelőiknek a további neveiként lelhetők fel, saját kóddal nem rendelkeznek. Így a `socket_set_timeout()` önmagában már nem létezik, csak a `stream_set_timeout()` egy álneve.
Ez nem csupán a PHP-ben fejlesztők munkáját könnyíti meg, de segíti a PHP további C-ben való fejlesztését is, hiszen így az újabb protokollok, adatforrások támogatásának bevezetéséhez nem kell teljes kezelőt írni, elég ennek a programozási felületnek a bővítményeit elkészíteni.
További újdonság, hogy a GD-támogatáshoz eleddig külön telepítendő libgd beszerzése nem szükséges, ugyanis a 4.3.0-s és későbbi kiadások már tartalmazzák. A különféle képformátumok kezelését biztosító külső könyvtárak fejlesztői változatára (libjpeg, libpng, libungif) viszont továbbra is szükség lesz. E fejlesztések mellett természetesen rengeteg hibát is kijavítottak benne, az Apache2 SAPI is közelebb került az éles kiszolgálókon való alkalmazhatósághoz. Ez véglegesen majd csak valamikor a nyárra várható, és a Zend2-motorra épülő PHP 5.0-val lesz hivatalosan is használható.

A jövő: PHP5 és Zend2

A történet 2001 nyarán kezdődött. Az akkori tervek szerint a Zend2-vel felszerelt PHP 5.0 várhatóan egyidőben jelent volna meg a 4.1-es változattal. Nos, azóta tudjuk, hogy nem így lett. A PHP 5.0-nak rengeteg elvárást kell kielégítenie, ennek megfelelően sok területen hatalmas előrelépésre lehet számítani.
A Zend-motor, ami a parancsfájlok feldolgozásáért felel, leginkább az objektumokkal kapcsolatos területen fog látványos eredményeket hozni. A PHP-t objektumtámogatottsága miatt sok vád érte, mivel a Java vagy a C++ képességeihez képest jócskán elmarad. Az egyik ilyen jelentős nehézség a jelenlegi 4-es változatokban az, hogy az „objektumváltozók” magát az objektumot tárolják, nem csak egy hivatkozást rá, mint ahogy az logikus lenne. Ennek következménye az, hogy minden egyes objektumpéldány egy másolat ahelyett, hogy ugyanazon objektumra mutatnának.
A Zend2-ben az objektumkezelés teljesen újra lett írva, ezáltal a fenti gondok megoldódnak, és ez utat nyit a komolyabb objektumalapú programozás felé. Hasonlóan nagy lépés lesz a kivételkezelés bevezetése. A `try`, `catch`, `throw` használatával immár a PHP is fel lesz vértezve azokkal az eszközökkel, amelyek segítségével „kiakadáskerülő” programokat építhetünk. ***Zeev Suraski***-nak, a Zend-motor egyik fejlesztőjének véleménye szerint a PHP ezzel a lépéssel valóban komoly, versenyképes objektumalapú nyelvvé növi ki magát. Az az előnye is meglesz a weblapba ágyazott Java lehetőségével szemben, hogy a PHP valóban pehelysúlyú parancsnyelv. Zeev mindezt röviden úgy foglalta össze, hogy a Java pont ilyen lett volna, ha parancsnyelvnek tervezték volna. Arra, hogy ténylegesen mik ezek az objektumkezelési újdonságok, egy későbbi cikkemben térek vissza.
További apróbb változások a karakterlánc-kezelésben várhatóak. Eddig is létezett az a lehetőség, hogy a karakterláncokat karaktereket tároló tömbként kezeljük, de ez egyrészt nem működik tökéletesen, másrészt a nyelvi szempontból teljesen eltérően kezelendő dolgokat jobb elkülöníteni. Ezért a jövőben a karakterláncok későbbi tömbként való kezelése figyelmeztető üzenetek megjelenését fogja kiváltani. A `$str[3]` helyett a jövőben a `$str{3}` forma használandó. A fejlesztők ennyivel természetesen nem érték be – ha már ilyen beavatkozás történik, érdemes kihozni belőle a legtöbbet.

Heiglig (Cece) Szabolcs (cece@php.net)
Veszprémben él. Hobbija, kedvenc időtöltése és munkája is a programozás. Szabadidejében hajlamos kerékpárra pattanni, vagy baráti társaságban szerepjátékokkal foglalkozni.

KAPCSOLÓDÓ CÍMEK

- http://www.zend.com/zend/future.php
- http://www.theopenenterprise.com/story/TEO2002120400001