Bemutatkozik a PEAR
A LEGO-zás művészete PHP nyelven (1. rész)

Hurrá, megérett a körte! Tudom, tudom, nyár eleje van, de mégis. A PEAR (PHP Extension and Application Repository, angolul a betűszó „körtét” jelent) ma már annyira népszerű, hogy nemrégiben belekerült a hivatalos PHP kiadásba is, azaz minden PHP fejlesztő számára alapértelmezetten rendelkezésre áll. Egy olyan közösségi projektről van szó, amelynek célja a nyílt forrású PHP bővítmények, gyakran használt alkalmazás-komponensek összefogása, kezelése és a fejlesztők rendelkezésre bocsátása.

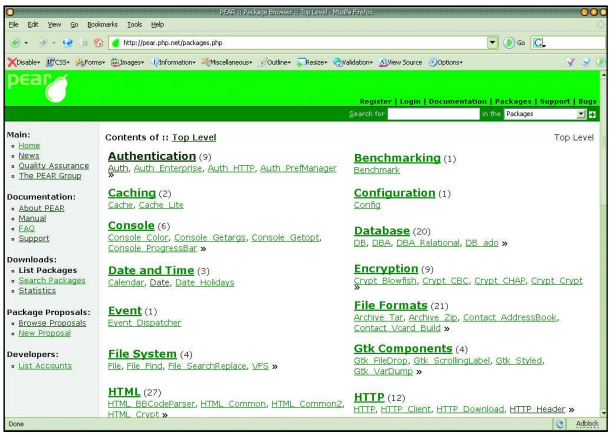
Amikor a rendszert használjuk, akkor egy csomagkezelő segítségével adott problémák megoldását célszó bővítményeket telepítünk a helyi gépre. Az összetevők eredetileg a ***PEAR*** kiszolgálóján egy központi tárban helyezkednek el. Ezeket a bővítményeket aztán a megfelelő ***PHP*** kódba beemelve (`include`) használhatjuk fel.
A ***PEAR*** elsősorban tehát egy olyan szervezett bővítményhalmaz, amelyhez a világ bármely fejlesztője hozzáférhet, szabadon felhasználhatja a saját programjában. Nem csak egy alkalmazás-tárról van azonban szó, hanem egy teljes körű, jól felépített, fejlesztő és felhasználóbarát, kényelmes keretrendszerről, amely nyílt forrású összetevőkön alapul. A ***PEAR*** felhasználók és a ***PEAR*** fejlesztők számára egyaránt nyújt felületet, szolgáltatásokat.
A most induló cikksorozatunkban először áttekintjük, hogyan épül fel a ***PEAR***, hogyan kell telepíteni, használni, milyen lehetőségek állnak rendelkezésünkre a keretrendszer használata közben, valamint szétnézünk a ***PEAR*** honlapján – amely szerves részét képezi a rendszernek. A későbbi epizódok során aztán sorra vesszük, hogy az egyes alkalmazásterületeken milyen megoldásokat kínál számunkra a ***PEAR***, és hogyan tudjuk ezeket használni.

A PEAR létjogosultsága

Mi, fejlesztők számtalan esetben kerülünk szembe jól ismert, gyakori problémákkal, amelyre a legtöbb esetben teljesen ugyanaz a megoldás. Ilyenek például az adatbáziskezelés problémája, levélküldés, matematikai függvények, és még sorolhatnám. Ravaszabb fejlesztőknek ilyenkor már eszébe jut, hogy feltehetően más is összefutott már a problémával, hátha megírta, és elérhetővé tette. Megkezdődik a komponensvadászat ***Google*** barátunk hatásos közreműködésével. A módszer hátránya a „pillanatnyiságában” rejlik: jó lenne a legmegfelelőbb megoldást megtalálni, népszerű összetevőt felhasználni, amelynek van támogatása is, az esetleges hibák kijavítása zajlik, s talán fejlődik is a program. Erről azonban nincs semmilyen biztosíték. Ezen kívül jó lenne egy egységes dokumentáció, és az sem ártana, ha nem kellene minden hasonló esetben fél napokat tölteni a komponensek keresésével. Szerencsénkre ezt a problémát már mások is felismerték, és létrehoztak egy keretrendszert a különböző megoldások szervezett összefogására, amelyet ***PEAR***-nek neveztek el.
Jelenleg 40 kategóriában 405 csomag áll a fejlesztők rendelkezésére, 215 fejlesztő dolgozik a csomagok naprakészen tartásán és továbbfejlesztésén, és a csomagokat együttesen eddig közel 12 milliószor töltötték le.

A PEAR alkalmazás-tárházának felépítése

A ***PEAR*** alapvetően a kiterjesztések és komponensek köré épül, amelyek hierarchiába szervezve, csomagok formájában találhatók meg a tárban. A csomagok hierarchiája valójában egy kategóriafa, ahol az egyes ágakban az azonos területre vonatkozó megoldások kapnak helyet. Egy-egy ilyen csomag magából a meghatározott felépítésű és stílusú programkódból és a hozzá tartozó ***XML***-alapú leíró információból áll. Minden egyes projekt önálló csomagot alkot, és minden projekthez tartoznak fejlesztők, dokumentáció, változatszám, kiadás, bejegyzés a ***PEAR*** weboldalán, stb.
A csomagok az előre meghatározott kódolási stílust, és az egyéb elhelyezési és névkonvenciókat követik egységesen. A komponensek csomagolását a fejlesztők végzik, akik egy új ***PEAR*** projekt regisztrálásával kezdhetik meg munkájuk közzétételét. Az egyes projektek más ***PEAR*** összetevők eredményeit is felhasználhatják, azaz hivatkozhatnak egymásra. Ezáltal egy függőségi viszony alakulhat ki az egyes csomagok között. Mint látható, nem csak felépítésében, de filozófiájában is rendkívül hasonlít a különböző ***Linux*** terjesztések csomagkezelő rendszereinek szerkezetére, még saját csomagkezelő szoftvere is van.



1. ábra A kategorizált csomaglist

A PEAR csomagkezelő

Nem csak a csomagok tárolására és a karbantartására nyújt megoldást a környezet, de lehetővé teszi a csomagok felhasználó- és gépbarát elérését is. A felhasználóbarát böngészés a ***PEAR*** weboldalán keresztül történik, a gépbarát megoldás pedig egy ***XML-RPC (XML alapú távoli eljáráshívás)*** megoldásra épülő, a helyi gépen futó csomagkezelő szoftver formájában testesül meg. Ez utóbbinak az a lényege, hogy az egyes szolgáltatások (a weboldalhoz hasonlóan) a ***PEAR*** kiszolgálóján futnak, s csak az eredmények jönnek át, természetesen a számítógép számára értelmezhető formában, kis túlzással olyan, mintha a felhasználói utasítások hatására a gép böngészné helyettünk a már sokat emlegetett weboldalt. A csomagkezelő egy az egyben a ***DEBIAN***-os ***apt-get*** csomagkezelő alkalmazás „kezelőfelületét” másolja – bár parancssoros alkalmazásról lévén szó, nem igazán beszélhetünk felületről... Ez a bizonyos alkalmazás teszi lehetővé a csomagok böngészését, helyi gépre történő telepítését, függőségek kezelését, majd a későbbiek folyamán azok karbantartását.

A PEAR környezet telepítése

A ***PEAR*** környezet a helyi gépen gyakorlatilag ezt az egy alkalmazást jelenti. A szoftver maga is egy ***PHP*** szkript, amelyet a ***PHP*** parancssoros értelmezője futtat (a héjprogramokkal azonos módon). Feladata a kapcsolattartás, a megadott csomag megkeresése és letöltése a ***PEAR*** tárból, majd annak kitömörítése és az előre meghatározott helyen történő elhelyezése. Ezt a bizonyos könyvtárat, ahová a helyben telepített alkalmazások kerülnek, el kell helyezni a ***PHP*** `include` elérési útvonalában, ami után a fejlesztőnek már csak annyi a dolga, hogy egy `include()` művelettel felhasználja a bővítményt.
A működéshez hasonlóan a telepítés is rendkívül egyszerű. A ***PHP 4.3.0*** változatától kezdve ez a bizonyos „manager” (amit mi csomagkezelőnek nevezünk) alapértelmezetten része a ***PHP***-nak, ha tehát van ***PHP***-nk, akkor van csomagkezelő is, amely a `pear` paranccsal indítható. Egy parancssoros programról van tehát szó, amely a debianos ***apt-get PHP***-s megfelelője. Ebben az esetben nincs semmi dolgunk.
Ha netán mi olyan ***PHP***-t használunk, amelyben ez nem szerepel (például a terjesztésben külön van választva), akkor telepítenünk kell a ***php4-pear*** vagy ***php5-pear*** nevű csomagot (a legtöbb rendszerben így hívják). Van kézi lehetőség is, de erre csak igen ritkán van szükségünk ***Linux*** alatt. Aki erről szeretne tájékozódni, az a ***http://pear.php.net/manual/hu/installation.getting.php*** címen olvashat róla.
Győződjünk meg róla, hogy a ***/etc/php[4-5]/php.ini*** fájlban az `include_path` változóban szerepel-e a ***PEAR*** helyi tár (általában a ***/usr/share/php*** könyvtár) elérési útja, ha nem, akkor adjuk hozzá, de közben vigyázni kell, hogy a mindenkori helyi könyvtár (.) is szerepeljen. Ez általában akkor fontos, ha még nem volt beállítva a változónak semmilyen érték, ekkor ugyanis az aktuális könyvtár az érvényes, de ha adunk meg értéket, akkor az alapértelmezett érték már nem érvényes.

Ismerkedés a csomagkezelővel

Aki nem ismeri az `apt-get`-et, annak álljon itt némi ízelítő, hogy hogyan kell használni. Az alapvető szerkezet:

```
pear művelet <csomagnév>
```

1. táblázat

Telepítés: `pear install <csomagnév>`	Megkeresi az adott nevű csomagot, letölti, kitömöríti a megfelelő helyre (függőségek esetén az összes többi szükséges csomaggal ugyanezt teszi)
Eltávolítás: `pear uninstall <csomagnév>`	Eltávolítja a megadott csomagot
Csomagok listája: `pear list-all`	Kilistázza az összes elérhető csomagot.
Telepített csomag összetevői: `pear list <csomagnév>`	Kilistázza egy már telepített csomag összes alkotórészét a teljes elérési útvonal feltüntetésével.
Információ egy csomagról: `pear info <csomagnév>`	Kilistázza az adott csomaghoz tartozó.
Keresés a csomagok között: `pear search <keresőszó>`	Visszaadja az összes olyan csomag nevét, amelyekre illeszkedik a keresőfeltétel (a nevében szerepel a keresőszó). Ha több szót is megadunk, azok között vagy kapcsolat áll fenn.
Csomag letöltése: `pear download <csomagnév>`	Letölti a csomagot (tar.gz formátumban) az aktuális könyvtárba, de nem telepíti azt.

Az első paraméter mondja meg, hogy mit szeretnénk, a második, hogy melyik csomaggal akarjuk ezt művelni (ha a művelet csomaghoz köthető). Ezek természetesen különböző kapcsolókkal tovább bővíthetők. A kapcsolók egy része a csomagkezelőnek szól, másik részével a műveletet pontosíthatjuk, de ezekre az egyszerűbb esetekben (ez a mostani helyzet is ilyen) nincs szükség. Nézzünk néhány egyszerűbb műveletet, amire szükség van (*1. táblázat*). Ezen kívül számtalan egyéb szolgáltatás áll még a rendelkezésünkre, amelynek nagy részét nem is fogjuk használni, ha nem teszünk közzé csomagokat, vagy nem kezelünk már meglévőeket.

Az olyan műveletek esetén, ahol az összes csomagra vonatkozó adatokról van szó (keresés, összes csomag megjelenítése, stb.) a teljes leíróadatbázis letöltődik a gépünkre. Az első keresés tehát a hálózat sebességétől függően elég lassú, de az utána következők már gyorsak, mert a csomagkezelő gyorstárazza az adatokat. Ez egyébként minden egyes csomagra igaz. Ha kérünk információt egy csomagról, az arra vonatkozó leíróinformáció tárolódik a helyi gépen. Ha közben megváltozott, akkor természetesen frissül on-line. Ezt az átmeneti tárat egyébként a pear clear-cache paranccsal lehet törölni, ez esetben egyébként a már letöltött csomagok (amelyek szintén megtalálhatók egy átmeneti tárban) is törlődnek.

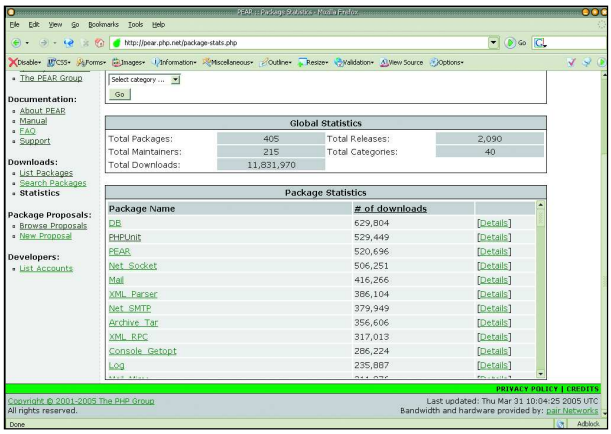
Többször belefutottam már abba a problémába, hogy az ilyen, teljes tárat érintő műveletek során hiba történt, hibaüzenetként pedig azt kaptam, hogy nincs elég memória. Ennek az az oka, hogy alapértelmezetten a *PHP* 8 megabájt memória lefoglalását engedélyezi, egy ilyen *XML* leíró struktúra viszont ennél többet kíván. A hiba úgy orvosolható, hogy a *PHP* parancssori felületének beállításaiban (*PHP5* esetében */etc/php5/cli/php.ini*) a `memory_limit` nevű változó segítségével megváltoztatjuk a memóriakorlátot. Hasonló problémával kerülünk szembe, ha túl sok a hálózati késleltetés, ezáltal a kód futása meghaladja a 30 másodpercet, ennyi ugyanis az alapértelmezett időkorlát *PHP* szkriptek futtatásakor. Bár ennek is és az előző memóriakorlát megemelésének is vannak biztonsági kockázatai, néhány esetben nem ússzuk meg, hogy ne változtassunk az értékeken.

A PEAR használata

Használat gyanánt (a környezet beállítása és a telepítés után) csak annyi a dolgunk, hogy a *PHP* szkriptben behúzzuk az adott fájlt. Lássunk egy példát erre is, Telepítsük, majd használjuk a *PEAR Date* csomagját, amelynek segítségével egy dátummal végezhetünk mindenféle varázslatos műveletet. Egy dátumot a `Date` osztály egy példánya reprezentál, és annak egyes tagfüggvényeivel vezérelhető meglehetősen egyszerűen.

Az első lépés: Adjuk ki rendszergazdaként a `pear install date` parancsot. Helyes működés esetén az alábbi kimenet keletkezik:

```
root@teve:/etc/php5/cli# pear install date
downloading Date-1.4.3.tgz ...
Starting to download Date-1.4.3.tgz (42,048 bytes)
............done: 42,048 bytes
install ok: Date 1.4.3
```



2. ábra A PEAR statisztika oldala

A második lépés: Írjuk meg az alkalmazást, amelynek kulcsmomentuma, hogy az első sorban szerepeltessük az `include('Date.php')` műveletet. A kód egy egyszerű példázatot szemléltetve így néz ki:

```
<?php

  require_once("Date.php");

  $date = new Date("1990-12-30");

  echo "A dátum évszáma: ";
  echo $date->getYear()."<br>";

  echo "Az évnek eme sorszámú napjára esik: ";
  echo $date->getJulianDate().".<br>";

  echo "Az évnek eme sorszámú hete: ";
  echo $date->getWeekOfYear().".<br>";

?>
```

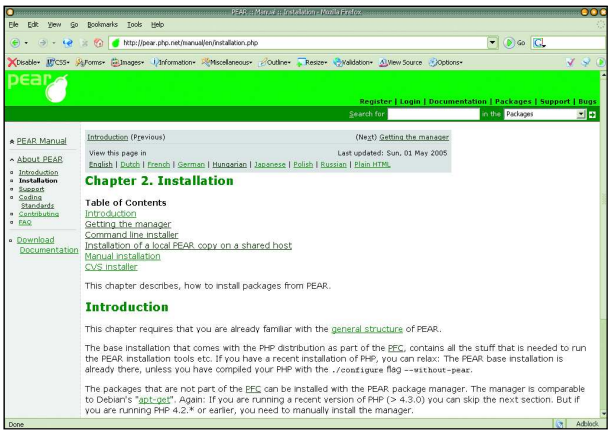
A PEAR csomagok leírása

Fejből nehéz volna kitalálni, hogy a fenti `Date` osztálynak milyen tagfüggvényei vannak, melyik mit csinál, és hogyan kell azokat használni. Ennek érdekében minden *PEAR* projektnek része egy *API* dokumentáció, és számos esetben tartozik hozzá végfelhasználói leírás is, amelyeket a *PEAR* honlapján, az adott projekt oldaláról lehet elérni. A fent tárgyalt esethez a dokumentáció a ***http://pear.php.net/package/Date/docs/1.4.3/*** címen érhető el.

A PEAR honlapja

Ha már szó volt róla: a honlap ugyanolyan szerves részét képezi a *PEAR* környezetnek, mint maguk a csomagok. Minden egyes csomaghoz felhasználóbarát, részletes információt találunk a ***http://pear.php.net*** címen. Az oldalon egyébként minden művelet elvégezhető, amelyet a parancssoros csomagkezelővel már megtanultunk, az csupán arra jó, hogy automatizált, és gyorsabban telepít, mint mi kézzel. A tájékozódás, csomagkeresés, részletes információk olvasása, stb. általában a webes felületen történik.

Minden projektnek van saját aloldala a *PEAR* honlapján.



3. ábra A PEAR felhasználói (fejlesztői) kézikönyv

Egységes kinézet és egységes tartalom jellemzi ezeket. Az egyes projektek legegyszerűbben a ***http://pear.php.net/packages.php*** címen érhetők el, ahol kategorizáltan böngészhetünk az egyes csomagok között. A csomag nevére kattintva juthatunk erre a bizonyos projekt oldalra, ahonnan akár le is tölthetjük a csomagot, és kézzel, úgynevezett félautomata módon is telepíthetjük. Ennek módja:

```
pear install <csomagnev>.tar.gz
```

A módszer előnye, hogy olyan helyen is telepíthetjük, ahol mondjuk nincs közvetlen internet-hozzáférés (ilyen lehet egy telep egyik csomópontja, és még sorolhatnám. Hátránya, hogy így nem kezeli a függőségeket a parancssori telepítő, azaz csak akkor megy fel gond nélkül a csomag, ha minden szükséges másik csomag is rendelkezésre áll.
Egyik hátránya az oldalnak, hogy a keresés során csak a csomag nevében kereshetünk (hasonlóan a parancssoros csomagkezelőhöz), a projektekhez tartozó hosszabb és rövidebb leírásban nem. Szerencsére egyelőre nincs olyan sok csomag, hogy ne lehetne átlátni azokat. Ez részben a kategorizált nézetnek köszönhető. A csomagok ugyanis a felhasználás illetve a technológiák szerint csoportosulnak, mi pedig általában tudjuk, hogy az adott probléma mely területhez kapcsolódik, és azonnal ott keressük.
Nem szabad megfeledkeznünk a letöltési statisztikákról. A ***PEAR*** ugyanis ezekkel méri az egyes csomagok népszerűségét. A ***http://pear.php.net/package-stats.php*** címen elérhető oldalon a csomagok aszerint vannak rendezve, hogy melyiket hányszor töltötték le. Ez persze nem pontos, de közelítőleg jó. Minél népszerűbb egy projekt, annál sokoldalúbb, és nagyobb a valószínűsége annak, hogy folyamatosan fejlődik és mindig karban van tartva. Adott esetben előfordulhat, hogy választanunk kell ***PEAR***-es és nem ***PEAR***-es megoldás között. A választást nagyban megkönnyíti, ha ismerjük a két csomag népszerűségét.
Mindezeken túl a ***PEAR***-ról minden információt megtalálhatunk az oldalon. Egy ***PEAR***-t gyakran használó programozó számára olyan a webapplikáció, mint a ***php*** oldala: élete jelentős részét tölti el annak használatával.

PEAR egyéb

A ***PEAR*** az bővítmény táron, a fejlesztők és felhasználók támogatásán túl egyéb feladatokat is ellát. A csomaghierarchia felsőbb szintjei önálló részhalmazokat képeznek. Ilyen részhalmaz például a ***PFC (PHP Foundation Classes)***, amelybe a mezei ***PEAR*** csomagoknál sokkal szigorúbb előírásoknak megfelelő csomagok kerülhetnek be. Csak stabil változatok lehetnek a ***PFC*** tagjai, azok közül is olyanok, amelyek zavartalanul képesek együttműködni másokkal azáltal, hogy szabványos alkalmazásfejlesztői felülettel (***API***) rendelkeznek. Mindemellett még teljesen általánosak is, azaz semmilyen környezethez sem kötődnek az indokoltnál jobban.
Egy másik ilyen részhalmaz (amely mára már külön projektté nőtte ki magát) a ***PECL (PHP Extension Community Library)***. Ez olyan ***C*** nyelvű bővítményeket, kiterjesztéseket tartalmaz, amelyek a ***PHP4*** részei. A ***PECL*** létrehozásának egyik motivációja az volt, hogy legyen hová átmozgatni a ***PHP*** részeként kezelt különböző bővítményeket. A legfontosabb különbség a ***PEAR*** és a ***PECL*** között az, hogy a ***PECL*** csomagok többnyire ***C*** nyelvűek, míg a ***PEAR*** csomagok ***PHP***-ben íródtak. A ***PECL*** csomagok tehát alacsony szintű, valódi bővítmények, amelyeket a ***PHP*** használ, és a fejlesztők a ***PHP***-n keresztül érhetik el. Tipikusan ilyen csomagok a különböző adatbázisokat kezelni tudó bővítmények. (***PHP4***-ben hasonló bővítmény végzi a ***MySQL***, ***PostgreSQL*** adatbázis kezelését is).
Ez a részhalmaz mára teljesen külön projektté nőtte ki magát, de továbbra is a ***PEAR***-től kölcsönzi szerkezetét.
A gyakorlatban ez annyit tesz, hogy a már ismertetett csomagkezelőn keresztül érhető el, a felhasználók számára áttetsző, hogy ***PEAR***, vagy ***PECL*** csomagot használnak.
A ***PECL*** csomagok telepítéséhez általában szükség van a PHP fejlesztői csomagjára, valamint komplett ***C*** fejlesztői környezetre (például ***C*** fordító, `make`), mivel a telepítendő alkalmazások a telepítés során helyben fordulnak le.
A projekt a ***http://pecl.php.net*** címen érhető el.
Mindezek mellett számos levelezőlista áll a fejlesztő közösség rendelkezésére, ahol tanácsokat kérhetnek a ***PEAR***-rel, csomagok használatával és minden egyébbel kapcsolatban.

Zárszó

Hatalmas előnyhöz juthat azzal a fejlesztő, ha a felmerülő problémákat okosan, időkímélő módon, kevés fáradsággal, mások kárán tanulva tudja megoldani. Még nagyobb az előnye abban az esetben, ha egy ilyen keretrendszer áll rendelkezésére ahhoz, hogy meg is találja a neki megfelelő komponenst. Továbbmenve az összetevő felhasználása is szervezettséget sugall: egy átgondolt, egységes rendszerben dolgozhat a tervező, fejlesztő. Ez a rendszer lehetővé teszi a folyamatos karbantartást, az egyszerű használatot, és az is rengeteget számít, hogy a csomagok mögött valóban áll egy szervezet, némi garanciáját nyújtva annak, hogy ez hosszú távon így is marad.
A sorozat következő részében a ***PEAR*** alapú adatbázis-kezelő megoldásokkal ismerkedünk meg, élén a legnépszerűbb csomaggal, a ***DB***-vel.

Komáromi Zoltán
(komi@kiskapu.hu)
25 éves, a BME hallgatója, mellette PHP-programozóként dolgozik. Kedvenc területe a multimédia.