PHP nyomkövetők (3. rész)

Sorozatunk előző részeiben áttekintést adtunk a Linux alatt elérhető PHP nyomkövetőkről. Szó esett a PHP nyomkövetés általános meggondolásairól, valamint beszéltünk a Gubed/Quanta nyílt forrású programok működéséről, beállításáról. Most a Nusphere::PhpED fejlesztőkörnyezetről és nyomkövetőről lesz szó.

Miért éppen a PhpED?

Ebben a részben kivételt tennék annyiban, hogy a *PhpED* program több platformra is létező implementációi közül nem a Linuxos, hanem a *Microsoft Windows*-ra írt változatot venném górcső alá. Ennek verziószáma jóval előrébb tart Linuxos társánál. Nehezen érthető számomra, hogy a *DBG* szerzője és a *NuSphere* vezetője, *Dmitri Dmitrienko* miért intézi ezt így. Azonban népszerűsége és használhatósága miatt nem hagyhatjuk a *PhpED*-et figyelmen kívül. E sorok nyomdába kerülésekor talán már a linuxos változat is eléri azt a 4.5-ös verziószámot, ami alapján e sorok születtek.

Ha az ember csak úgy vaktában elkezd *PHP* nyomkövetőt vagy profilkészítő alkalmazást keresni, mert eddig még nem használt semmi ilyesmit, de a baj most erre kényszeríti, akkor nagy valószínűséggel ez az eszköz akad a kezébe. Közkedveltsége abból is fakad, hogy van benne *FTPS (TLS/SSL)*, *WebDAV/HTTPS (SSL)* és *SOAP* kliens támogatás, tud kapcsolódni *SQL* szerverekhez, igen jó a dokumentációja, s hogy a támogatói levelezőlistán maga a szerző válaszol meg szinte minden levelet (ami őt érinti). Olvasható is vele egy interjú a ➲ *www.nusphere.com* oldalon. Innen tölthető le maga a program is.

PhpED fejlesztőkörnyezetet IDE, de gyorsan!

Tegyük fel, hogy úgy áll a helyzet, hogy van egy (távoli) webszerver, amin ott vannak a *PHP* anyagaink, fájljaink. Semmi előzményünk nincs semmilyen fejlesztőkörnyezetben, csak nyomkövetni akarunk, de még tegnap, ha lehet. Ekkor hogyan érdemes elkezdeni a munkát a *PhpED*-del?

A webszerver okosítása

Először is a webszerverben futó PHP-értelmező verziószámával egyező számú *dbg__.so*-fájl másolandó fel a *PhpED* kliens oldali könyvtárából a szerver *extension_dir* könyvtárába (aminek tényleges mivoltát pl. egy *phpinfo()*-ból, vagy magából a *php.ini*-ből tudhatunk meg). A *PHP*-értelmező verziószámát a `php -v` parancs is megmutatja.

A *php.ini*-be be kell írnunk néhány paramétert. Melegen ajánlott, hogy csak a *localhost*ot engedjük:

```
extension=dbg.so
[debugger]
debugger.enabled=on
debugger.profiler_enabled=on
debugger.hosts_allow=localhost
debugger.hosts_deny=ALL
debugger.ports=7869, 10000/16
```

Amiatt lehetséges a szerveren a *localhost* (helyi gép) használata, mert majd a kliensünk operációs rendszeréből indítunk egy ssh-alagutat a webszerver felé, és onnantól kezdve az „helyinek” fog látszani:

```
"c:\program files\putty\putty"
➥-ssh -R 7869:localhost:7869
➥login@szerverneve
```

A webszerver újraindítása után (sajnos rendszergazdaként lehet csak ilyet művelni, vagy – egy jól beállított rendszeren – az erre kiválasztottak ezt `sudo`-val is megtehetik) érdemes kiadni egy `php -m` parancsot, ami a szervermodulokat mutatja. A lista végén látszania kell a nyomkövető protokollért felelős modulnak:

```
[Zend Modules]
DBG
```

A PhpED beállítása

Abban az esetben, ha tényleg szinte semmit nem akarunk, csak hibát keresni, akkor is meg kell néhány információt adnunk a fejlesztőkörnyezet számára:

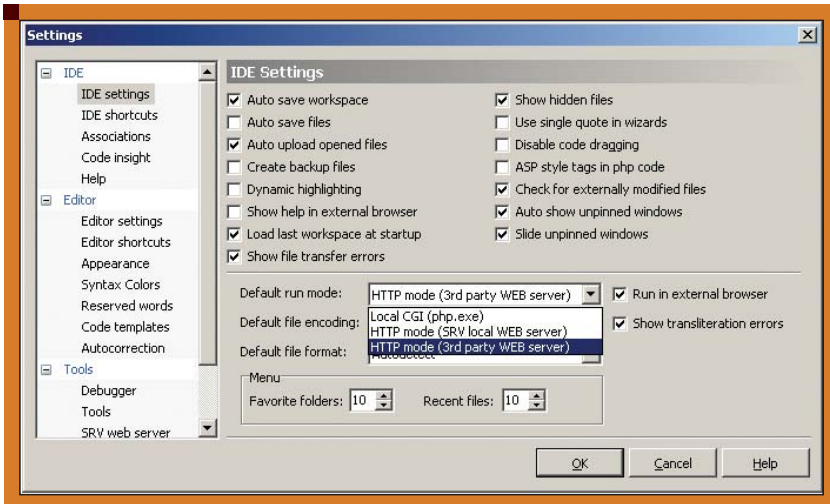
- Új munkaterületet (workspace) nevezünk ki, mondjuk „szimpla” néven
- Új projektet is gyártunk, melynek a gyökérkönyvtárát például „minimalis”-ra nevezzük. Ilyenkor még meg kell adni legalább a „Mapping root URL”-t és a webszerver document gyökérkönyvtárát, például *http://szerverneve/egyeb* és */var/www/html/ez_az_a_resz/*
- Ekkor feljön egy dialógusablak, amiben végre elmenthetjük a projektünket például *minimalis.ppj* néven. Érdemes ügyelni arra, hogy a projekt gyökérkönyvtárának a neve is emlékeztessen erre a névre.

A nyomkövetés indítása

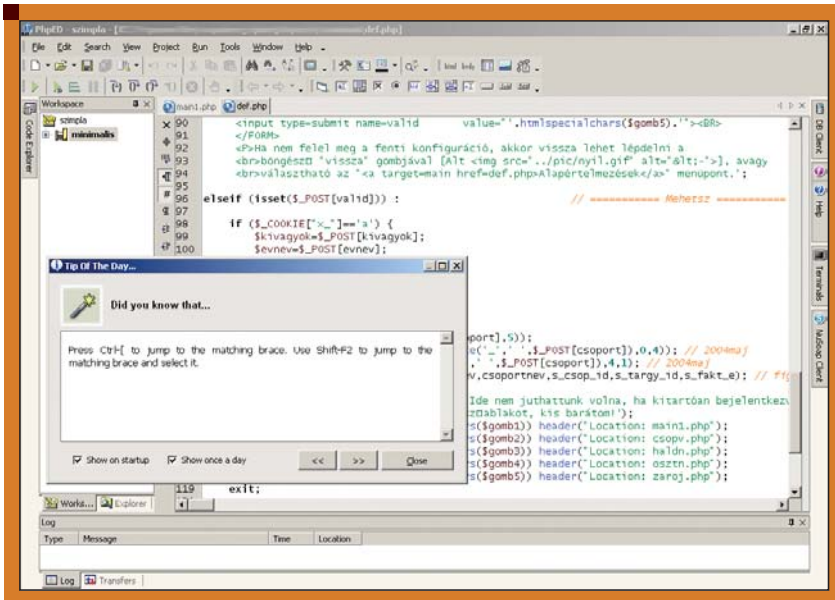
Ezek után már nyithatjuk is a vizsgálni kívánt URL-t: A *Tools* menüből az *"Open URL"*-be írva:

http://szerverneve/admin

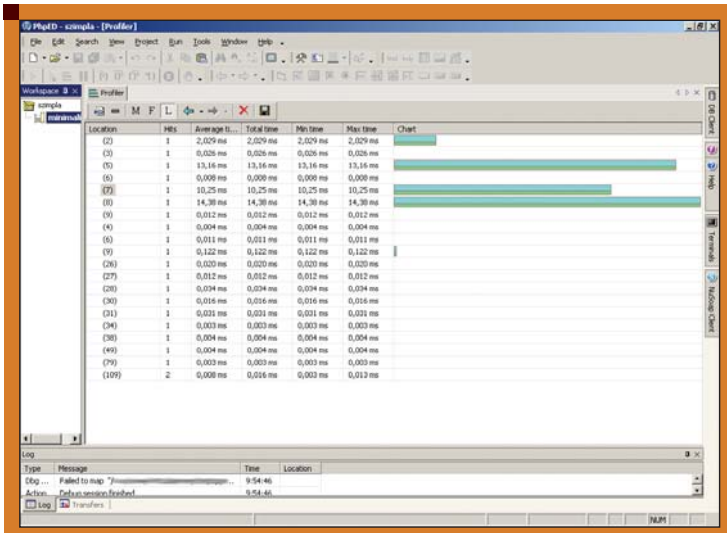
Itt van lehetőség eldönteni, mely jelölőnégyzetbe kattintunk pipát: nyomkövetni akarunk-e (***Run debug session***), teljesítményt szeretnénk elemezni (***Run profiler session***), esetleg külső böngészőt szeretnénk használni (***Run in external browser***).
És már megy is a nyomkövetés. Lehet lépegetni sorról sorra. Az első soron automatikusan megáll a nyomkövetés – ennek megváltoztatásához van is egy jelölőnégyzet a beállításoknál. Állíthatunk töréspontot, feltételhez kötve is. Lehet nézni a változók értékét (egérrel rámutatva), vagy akár változtatni rajtuk.
A beépített profilkészítő (*3. ábra*) bosszantó módon csak 20 sort mutat az ingyenesen kipróbálható változatban, úgyhogy lényegében felesleges is sokat beszélni róla, annyi más szabad eszköz van e célra. Ha valakinek mégis kell a ***PhpED*** teljesítményelemzésének eredménye, el lehet menteni egy ***.xml*** fájlba (floppi ikon). Ha a teljesítményelemzés közben szeretnénk megnézni azt a kódrészletet, aminek a mérési adatait láthatjuk, akkor szükség van még valamire, amit könytárleképezésnek (***mapping***) hívnak, és ami a forráskód beazonosítására szolgál. Ez persze nem árt a normál nyomkövetéshez sem; akkor ugyanis nem ad figyelmeztetéseket az *„**unmapped files**"*, azonosítatlan fájlok miatt.



1. ábra Beállítások



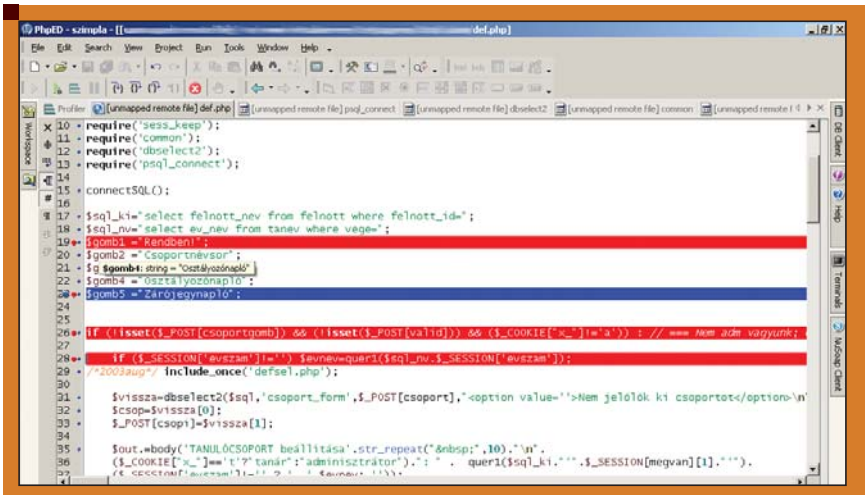
2. ábra Tipp mutatásával indul a PhpED kliensprogram



3. ábra A profilkészítő

Könytárleképezés után szebb az élet

Ez nem egy triviális feladat. Nem véletlen, hogy a ***PhpED*** levelezőlistáján minden második kérdés erre vonatkozik. Eleinte nem volt világos, hogy mit mivel akar összepárosíttatni a program. Eleinte azt a két információt adtam meg csak, hogy mi a webes URL és a webszerver fájlrendszerében való elérési út. Kiderült azonban, hogy csak akkor működik megfelelően



4. ábra A PhpED teljes díszben

a program, ha a helyi gépen is megtalálhatóak az érintett fájlok. (Vagy én nem találtam meg a módját a másképp szervezésnek; mindenesetre a nagy vetélytárs, a *Zend Studio* jobban viselkedett kezeim között e tekintetben.) Magyarán a projekt tulajdonságait beállító fő ablaknak három (nem egymás közelében levő) sora tartozik össze: A fent említett webes URL és a webszerver fájlrendszerében való elérési út mellett a *„Projekt”* rész alatti *Root directory*-t (gyökérkönyvtárat) is be kell állítani úgy, hogy onnan kiindulva azonos elérési úttal lehessen itt megtalálni a *.php* fájlokat, mint a webszerveren. S ekkor már a profiler is meg tudja nevezni a forrásokat, és rá tud állni az általunk keresett részre.

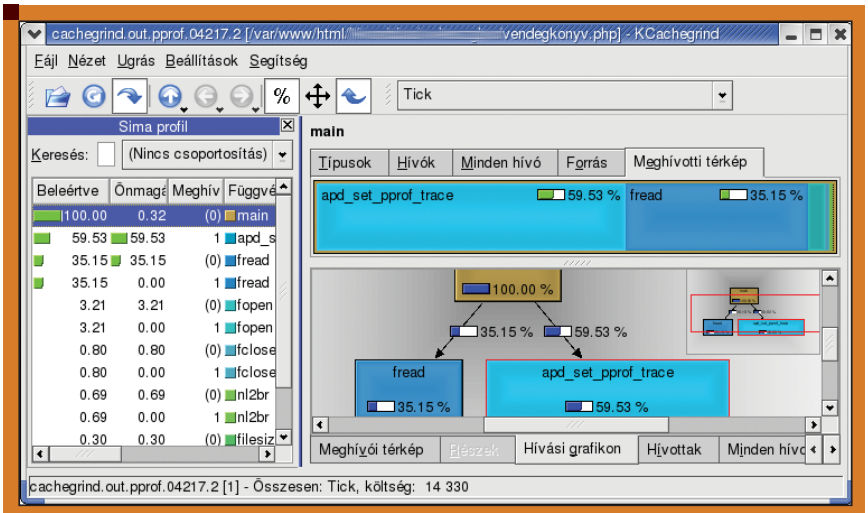
Ha 20 sornál hosszabb a kód: ne APD fel!

A *PhpED* teljesítményelemzése csak 20 sort mutat. Azonban aggodalomra nincsen ok: rendelkezésünkre áll az *Advanced PHP Debugger* (*APD*) ➲ *pecl.php.net/package/apd*. Neve megtévesztő. Szerzőjével, *George Schlossnagle*-vel váltottam egy e-mailt, amiben megkérdeztem, milyen klienst tud ajánlani programjához. Azt írta, hogy programja igazából teljesítményelemzésre való – nyomkövetéshez használjam az *Xdebug*-ot. Próbáltam őket (az *APD*-t és az *Xdebug*-ot, amiről a következő részben lesz szó) egyszerre beintegrálni a webszerverbe, de ilyenkor sajnos elszállt a webszerver *segfault* hibával, ami nem is csoda, hiszen ez a modul is módosítja a *PHP* belső függvényhívásait, így más kiterjesztéseket zavarhat. Az *Xdebug* behúzását tehát „kikommenteztem” (pontosvesszővel) a *php.ini*-ben. *„PHP Performance Profiling”* (*PHP* teljesítményelemzés) címmel olvasható angol nyelven egy részletes írás az *APD*-ről: ➲ *www.linuxjournal.com/article/7213*, valamint a ➲ *www.php-editors.com/php_manual/ref.apd.html* is jó eligazítást ad. Az *APD* inkább azt tűzte ki céljául, hogy olyan profilkészítő lesz a PHP számára, mint amilyen a *gprof* a *C* nyelv számára, vagy a *Perl Devel::DProf*-ja.

Az APD forrásból fordítása a `phpize; ./configure; make` parancssorral történhet. A fordításhoz kell néhány csomag a webszerverre, ha még nincs ott: d*pkg* alapú rendszereknél az `apt-get install php-dev php-devel automake gcc cpp` parancssor segít sokat. Ezeket a fordítás után `apt-get remove`-val érdemes eltávolítani, ne éktelenkedjenek ott feleslegesen (és a biztonságot aláásva). A keletkezett *modules/apd.so* fájl bemásolandó a fent tárgyalt *„extension_dir”* könyvtárba (így tud majd betöltődni, mint *„zend_extension”*). A *php.ini* fájlba még be kell írnunk e sorokat, majd újraindíthatjuk a webszervert:

```
zend_extension = /az/apd.so/
➥elérési/útja
apd.dumpdir = /a/kimeneti/
➥fájlok/könyvtárának/neve
apd.statement_trace = 0
```

Ez utóbbi a soronkénti nyomkövetést állítja. 1-esre állítva igencsak lelassul az alkalmazás. A középső sorból az látszik, hogy meg kell adni egy kimeneti könyvtárat (amit aztán célszerű pl. egy `chmod 777` paranccsal írhatóvá tenni) – ide fognak érkezni az *APD* által gyártott kimeneti fájlok, például a *pprof.04214.0.* (A fájlnév végén levő számok processzazonosítóra utalnak.) Az *APD* lefordítása után találunk a könyvtárban egy *pprofp (PHP profiler parser)* nevű, parancssorból futtatható szkriptet, sok kapcsolóval (mi szerint rendezze az adatokat,



5. ábra APD-kimenet kcachegrind ruhában

mit mutasson és mit ne stb.). Az APD kimenetéből tehát messze nem pusztán az egyes lépések futási ideje derül ki, hanem lehet pl. függvényhívási fát is kérni.
Dpkg alapú rendszerekben a ***kdesdk*** csomag része a ***pprof2calltree***, amivel a a ***kcachegrind*** számára feldolgozhatóvá tehetjük az APD kimenetéből kapott fájlokat.
A ***kcachegrind*** (és tartozéka, a ***callgrind***) letölthető innen:
➲ ***kcachegrind.sourceforge.net***.
A teljesítményelemzésnek alávetendő PHP szkriptünk elejére illesztendő az `apd_set_pprof_trace();` parancs. Ha megkapja a PHP-értelmező az adott szkriptet, elkészül a ***pprof-kimeneti-fájl*** az ***apd.dumpdir***-be. Ezek után egy jól célzott

```
pprof2calltree -f  pprof- kimeneti-fájl
```

paranccsal előállíthatjuk a ***cachegrind.out.pprof-kimeneti-fájl***-t, amit már átadhatunk a ***kcachegrind***-nek. Az eredményül kapott szép (interaktív) felületen vizuálisan láthatjuk, mire mennyi erőforrás kell (s azt is, hogy a profilkészítés bizony erősen használja a gépünket).
Jó tudni, hogy általában a ***.kde/share/config/kcachegrindrc***-ben vannak eltárolva a legutóbbi nézethez tartozó beállítások.
A grafikonban alul (kissé elrejtve) található a függvényhívási grafikon megjelenítőgombja. A forrásoknak a helyi gépen is megtalálhatóaknak kell lenniük a profil grafikonjának elkészítésekor, ha a kódba is bele akarunk látni.
Ha valódi, éles weboldalakat szeretnénk mérni, akkor minél többféle módon érdemes azt megtenni, különböző terheltségek mellett, és több adatsort átlagolni a hiteles válaszhoz. Amit az APD (és más profilkészítők) tudnak mérni, az a valós idő, nem pedig a processzoridő (hiszen többfeladatos rendszerekben más programok is ténykednek a processzoron), és attól is eltér, amit a felhasználó átél, amíg megkapja a kért weboldalt. Ha csak úgy odaírjuk az `apd_set_pprof_trace();` parancsot egy PHP fájl elejére, gondot okozhat, hogy sok fájl keletkezik az éles környezet számítógépén. Ettől le is lassul a rendszer. Emiatt érdemes egy feltételhez kötni a profilkészítési kérést a mérendő PHP fájlunkban:

```
$DEBUGIPS = array('84.2.93.34',
➥'192.168.0.13');
if(array_search($_SERVER [REMOTE_IP],
➥$DEBUGIPS)) {apd_set_pprof_trace();}
```

Azaz csak a kiválasztott IP-címekről érkező kérések esetén indul el a mérés.
Következő részünkben két ***Xdebug***-alapú PHP-nyomkövetőt tekintünk meg, az ***ActiveState::Komodo*** és a ***Xored::TruStudio*** fejlesztőkörnyezeteket.

Szabó Zoltán
(szz@freemail.hu)
Négy gyermekével és feleségével Pannonhalmán él. Tíz éve kísérletezik a Linuxszal. Matematikát és informatikát tanít, diákotthonban keseríti a rábízottak életét. Szívügye a PHP, a PostgreSQL és a Moodle.