Webfejlesztés PHP 4.0 és FastTemplate 1.1.0 segítségével

Ismerkedjünk meg egy időt megtakarító megoldással: a FastTemplate 1.1.0-val!

A webfejlesztésről közismert, hogy nagyon nehéz lépést tartani az általunk használt programeszközök hihetetlen gyors fejlődésével. Egy évvel ezelőtt még csak nem is hallottam a PHP nevű parancsnyelvről, ma pedig teljes munkaidőmben PHP-programozással foglalkozom.

Két csapatunk van: az egyik kizárólag PHP-programokat ír RedHat Linuxos munkaállomásokon, a másik csak DreamWeaverrel dolgozik NT-alapú PC-ken. A marketingrészleg meghatározza az oldalak célcsoportját; a grafikuscsapat pedig az adatok alapján kialakítja a HTML-lap látványvilágát. Ezután a programozóké a főszerep, hogy a PHP képességeit kihasználva egységessé kovácsolják a HTML-lapokat, és az aktív adatokat beillesszék a lapvázlatokba.

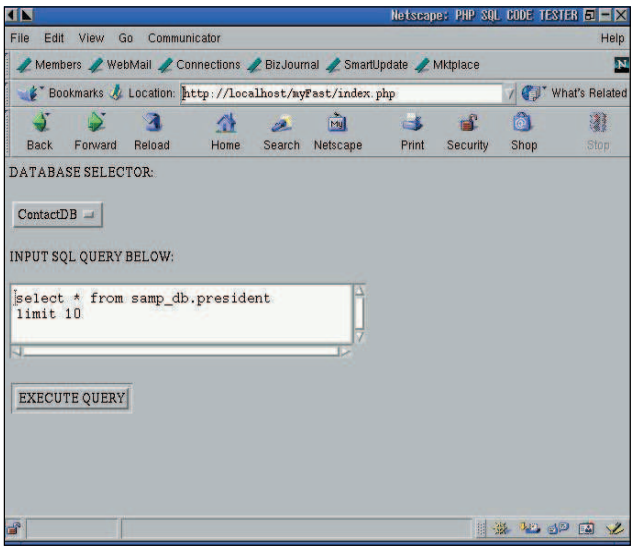
Mindez ragyogóan működik – egészen addig, amíg valaki mondjuk, meg nem akarja változtatni egy bizonyos gomb feliratát 46 különböző oldalon. Ez ugyanis azt jelenti, hogy 46 fájlban 46 változtatás szükséges csak azért, hogy egyetlen kis gomb másképpen nézzen ki! De hála az égnek, létezik a grep!

Az előbbi gondot azonban el is kerülhetjük, ebben kívánunk segítséget nyújtani ezzel az írással. A HTML-sablonok révén a fent említett gomb egyetlen fájl megváltoztatásával átalakítható lenne, ugyanis a sablonok használata lehetővé teszi, hogy a PHP-kódot elválasszuk a HTML-től. A PHP-kód módosítása nem lesz hatással a HTML-re, és fordítva.

A minták a következőképpen működnek: a PHP-program külső fájlokat nyit meg, és értelmez, amelyek a HTML-laphoz tartozó kódot tartalmazzák. A HTML fájlok az adatok helyén helyőrző jeleket (placeholders) tartalmaznak, amelyek futásidőben helyettesítődnek be. Ezek a tagok többnyire valahogy így néznek ki: `{ITEM}`. Ha a PHP-program felfedez egy tagot, behelyettesíti a megfelelő adatot, ezáltal a PHP kódfájlok és HTML fájlok teljesen elválasztva tárolhatók, így a programozók és a tervezők nem zavarják egymás munkáját.

Több nyílt forráskódú mintacsomag is elérhető, ezek között akadnak kicsit bonyolultabbak és hatékonyabbak is. Létezik azonban egy csomag, amelyben a végrehajtási sebesség és a használhatóság szerintem megfelelő egyensúlyban van, nevezetesen a CDI FastTemplate 1.1.0 programja. Szerzője, *Joe Harris* a General Artistic License felhasználási szerződés alapján adta ki művét. Joe a leírásban azt is közzéteszi, hogy a FastTemplate 1.1.0 tulajdonképpen egy azonos nevű Perl-modul PHP-változata. Az eredeti Perl-modul *Jason Moore* munkája.

A FastTemplate alapötlete: a honlapot szétbontjuk alapelemekre – egyedi gombokra, jelölőnégyzetekre vagy akár szövegsorokra. A táblákat fejlécre és sorokra kell törni, a sorokat pedig cellákra bontjuk. Minden elemhez saját HTML fájlt rendelünk, amelyben statikus (állandó) kód és egy, a változó adatot jelképező különleges tag található. A honlap ezekből az egyszerű elemekből épül fel. Végül egy olyan fájlt kapunk, mint amilyen a példában szereplő main.tpl, amely egyetlen `{BODY}` tagot tartalmaz. Néhányan ugyan jobban kedvelik a `{CONTENT}` elnevezést; de tulajdonképpen nem a név számít. Ezt a legfelső szintű tagot a teljes lap adataival fogjuk helyettesíteni. Ha szükséges, a honlap minden egyes lapjához létrehozhatunk egy-egy legfelsőbb szintű fájlt, melynek módosításával a teljes honlap kinézetét meg lehet változtatni. A példaprogramban is így fogunk eljárni.



1. kép Próbaoldal SQL lekérdezéshez

Lássuk, milyen előnyök származnak az eddig elmondottakból! Mivel a lapon található összes elemhez (widgethez) mintafájlokat szeretnénk készíteni, a minták használatával minden egyes lap tartalmának az utolsó bitig a helyén kell lennie. Ez jó, hiszen rákényszerít arra, hogy mérnökként tervezzünk, a meggondolatlan programozók életét viszont megkeseríti. Egy újabb lehetőség hozzáadása ezután nemcsak egyetlenegy, hanem legalább három fájl megváltoztatását igényli. Honlapunk ezáltal jobbá válik, hiszen tervezése átgondoltabb, részletesebb és finomabb lesz.

Elsőre talán úgy tűnhet, hogy rengeteg apró mintafájlt fogunk felhalmozni. Ez igaz is, de a mintafájlok széles körben újrafelhasználhatók. Ahogy az alkalmazás mérete növekszik, úgy egyre kevesebbet kell felhasználni. Akár több adattag is elhelyezhető egyetlen mintafájlban, ez csökkenti ugyan ezek számát, de egyszersmind kisebb mértékben lesznek újra felhasználhatóak.

A FastTemplate értékeléséhez látnunk kell működés közben, valamint azt is, hogy mire képes a weblap karbantartása során. Tételezzük fel, hogy van elérésünk egy olyan linuxos számítógéphez, amelyen már fut az Apache, a PHP és valamilyen relációs adatbázis-kezelő (én MySQL-t használtam), és rendszergazdai jogosultsággal rendelkezünk.

A FastTemplate telepítése pontosan olyan könnyű, mint amilyennek látszik. A honlap a ➲ http://www.thewebmasters.net/php/FastTemplate.phtml címen érhető el. Töltsük le a FastTemplate-1_1_0.tar.gz fájlt amennyiben PHP 4.0-t használunk, egyúttal a diff fájlt is (php4.diff). Mozgassuk a letöltéseket a dokumentumgyökérbe és az `untar` parancs segítségével csomagoljuk ki.

Ez létrehoz egy FastTemplate/ nevű könyvtárat. Ha PHP 4.0-t használunk, helyezzük a php4.diff fájlt a FastTemplate/-be és futtassuk le a következő parancsokat:

1. lista mysql.php

```
<%/////////////// index.php ///////////////////////

// ha valaki a EXIT PROGRAM-ra kattint,
// megmutatjuk a kilépő képernyőt, és kilépünk

if (isset($goodbye)) {
   include ("./goodbye.php");
   }

include ("class.FastTemplate.php");
$tpl = new FastTemplate(".");

$tpl->define (array(main     => "main.tpl",
                    form     => "form.tpl",
                    select   => "select.tpl",
                    option   => "option.tpl",
                    submit   => "submit.tpl",
                    textarea => "textarea.tpl"));

$tpl->assign (array(TITLE    =>
             ➥"PHP SQL CODE TESTER",
              FORM_ACTION    => "mysql.php",
              FORM_METHOD    => "post",
              STRING1        =>
             ➥"DATABASE SELECTOR: ",
              STRING2        =>
             ➥"INPUT SQL QUERY BELOW: ",
              SELECT_NAME    => "database",
              SELECT_SIZE    => "1",
              SUBMIT_VAL     => "EXECUTE QUERY",
              TEXTAREA_NAME => "query",
              TEXTAREA_COLS => "40",
              TEXTAREA_ROWS => "3" ));

                    //Feltételezzük a MySQL
                    //használatát.
$host="localhost"; //Módosítsuk megfelelően
$user="bill";      // e három sor
$password="megan"; // beállításait.

mysql_connect($host, $user, $password);
             ➥$db_table = mysql_list_dbs();

//szerezzük meg a helyi adatbázisok nevét, és
// csatoljuk a selector lehetőséghez.

for ($i = 0; $i < mysql_num_rows($db_table);
$i++) {
    $optionval = mysql_tablename($db_table, $i);
    $tpl->assign(array(OPTION_TAG
                ➥=>"$optionval"));
    $tpl->parse(OPTIONS, ".option");
    }

$tpl->parse(SELECT, "select");
$tpl->parse(TEXTAREA, "textarea");
$tpl->parse(SUBMIT, "submit");
$tpl->parse(BODY, "form");
$tpl->parse(MAIN, "main");

$tpl->FastPrint();
exit;
%>
```

```
patch class.FastTemplate.php3 php4.diff
mv class.FastTemplate.php3 class.FastTemplate.php
```

Ezután mozgassuk át a class.FastTemplate.php fájlt a PHP include könyvtárunkba. Ha nem vagyunk biztosak benne, hogy hol található, ellenőrizzük a php.ini fájlt (alapértelmezett esetben a /usr/local/lib/-ben helyezkedik el). Keressük az `include path` beállítást!
A class.FastTemplate.php fájlt közvetlen módon, az `include( )` paranccsal is csatolhatjuk a PHP-programokhoz, ez azonban többletmunkával jár, én is többnyire megfeledkezem róla.
Készítsünk valahol a dokumentumgyökérben egy másik könyvtárat a példafájloknak! Ehhez szükségünk lesz az *1.* a *2.* és a *3. listában* található programokra. A 4. listát egyedi mintafájlokká kell széttördelni (lásd ➲ ftp://ssc.com/pub/lj/listings/issue86/).
A példa rövidsége és egyszerűsége ellenére jól mutatja, hogy a FastTemplate könnyedén képes olyan táblákat is kezelni, amelyeknek sorai és oszlopai egyaránt dinamikusan változhatnak. Az oldal igazából csak annyira lesz bonyolult, amennyire az oldalon lévő elemek bonyolultak.
Vessünk egy pillantást az első listára! Ahhoz, hogy használhassuk a FastTemplate-et, egyszerűen csak csatoltuk a class.FastTemplate.php fájlt, és létrehoztunk egy példányt. A példányosítóban látható `(".")` azt jelöli, hogy mintafájljaink a jelenlegi könyvtárban találhatók, ezek azonban akárhol lehetnének. A FastTemplate osztály négy fő eljárással bír: `define( )`, `assign( )`, `parse( )` és `print( )`.
A `define( )` eljárás a programunk által használt külső fájlneveket címkékhez csatolja. Megjegyzendő, a mintafájloknak nem kötelező tpl-re végződniük, és az is elképzelhető, hogy egy programban mindössze egyetlen `define( )` hívásra van szükség.
Az `assign( )` eljárás a `{}` jel nélküli adattagokat rendeli ahhoz az adathoz, amivel helyettesíteni akarjuk. Egy programban több `assign( )` hívás is lehet. Kezdjük mindig a legkisebb, legalapvetőbb összetevővel, és építsünk belőle egyre nagyobbat, bonyolultabbat. Én a `parse( )` eljárást értettem meg a legnehezebben. Ugyanis ez az eljárás értelmezi a mintát, végrehajtja az adattag-behelyettesítést, és az eredményt egy helyi változóban tárolja. Ez a helyi változó a `parse( )` első értéke. A második érték az a fájlkezelő (file handle), amit a `define( )` eljárással megadtunk. Ha a `parse( )` második értéke `"."` jellel kezdődik, akkor az új érték az előző végéhez kapcsolódik (amennyiben létezik előző érték). Ebben a példában ezt a módszert használjuk arra, hogy létrehozzuk az adatbázis-tartalomjegyzéket. A PHP-program igazi erősségét használjuk fel arra, hogy a megjelenítendő adatot betöltsük vagy kiszámítsuk.
Végül a `print( )` alapértelmezés szerint kiírja a legutolsó parse-eljárás eredményét. Természetesen egy korábban értelmezett eredményt is átadhatunk értékként. A lapon egynél több `print( )` is lehet, sőt néha ez az egyszerűbb módja a HTML-oldal megjelenítésének. Általában az oldal létrehozására több lehetőségünk is van. Amelyik működik és hatékony, az jó.
Akkor lássuk a példát! Ha a böngészővel betöltjük az első lista példáit, az *1. képhez* hasonló képernyőképet kell látnunk. Válasszunk egy adatbázist a Selectorban, gépeljünk be egy lekérdezést,

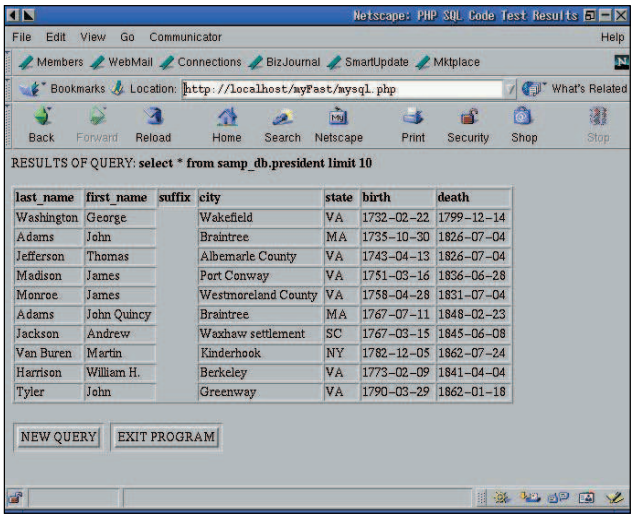
2. lista mysql.php

```
<%/////////// mysql.php ///////////////////////
include ("class.FastTemplate.php");
$tpl = new FastTemplate(".");
$tpl->define(array( main   => "main.tpl",
                    body       => "body.tpl",
                    table      => "table.tpl",
                    headerdata => "headerdata.tpl",
                    row        => "row.tpl",
                    rowdata    => "rowdata.tpl",
                    submit     => "submit.tpl",
                    form2      => "form2.tpl"));
$tpl->assign(array( TITLE     =>
                    ➥"PHP SQL Code Test Results",
                    SUBMIT_NAME => "SUBMIT",
                    SUBMIT_VAL  => "NEW QUERY",
                    FORM_ACTION => "index.php",
                    STRING1     =>
                    ➥"RESULTS OF QUERY: ",
                    QUERY       => "$query",
                    FORM_METHOD => "POST"));
                    //MySQL-t használunk.
$user="bill";       //Ebbe a három sorba
$host="localhost"; //saját beállításainkat
$password="megan"; //írjuk be.
mysql_connect($host,$user,$password);
mysql_select_db($database);
if(!strcmp($query, "")){
//ha nincs lekérdezés, állítsunk be egy
//alapértelmezettet
   $query = "SHOW TABLES";
   }
$query = stripSlashes($query);
$result = mysql_query($query);
//adjuk a fejléccellákat a fejlécsorokhoz.
for ($i = 0; $i < mysql_num_fields($result);
                                    ➥$i++) {
   $thval = mysql_field_name($result,$i);
   $tpl->assign(array(HEADERDATA => "$thval"));
   $tpl->parse(HEADER, ".headerdata");
   }
// adjuk a sorcellákat a sorokhoz, majd
// adjuk a sorokat a testhez.
for ($i = 0; $i < mysql_num_rows($result); $i++) {
   $row_array = mysql_fetch_row($result);
   for ($j = 0; $j < mysql_num_fields($result);
                                    ➥$j++) {
   $trval =  $row_array[$j];
   $tpl->assign(array(ROWDATA => "$trval"));
   $tpl->parse(ROW, ".rowdata");
   }
   $tpl->parse(ROWS, ".row");
   $tpl->clear("ROW");  //<- tegyük el
                        //megjegyzésbe ezt a
                        //sort és figyeljük meg
}                       //az eredményt.
$tpl->parse (TABLE, "table");
$tpl->parse (SUBMIT, "submit");
$tpl->assign(array(SUBMIT_NAME => "goodbye",
      ➥SUBMIT_VAL  => "EXIT PROGRAM"));
$tpl->parse (GOODBYE, "submit");
$tpl->parse (FORM2, "form2");
$tpl->parse (BODY, "body");
$tpl->parse (MAIN, "main");
$tpl->FastPrint();
%>
```

üssük le az EXECUTE gombot, és a lekérdezés eredménye máris megjelenik (lásd a *2. képet*).

Most lássuk a grafikai tervező szerepét! Az a feladatunk, hogy a teljes alkalmazás látványát megváltoztassuk. Képzeletbeli főnökünk éppen most mondta, hogy a szürke hátteret le kell cserélni sárgára. Továbbá azt szeretné, ha minden középre lenne rendezve, nem pedig bal oldalra. Ja igen, és az összes szöveg zöld színű legyen! Egy FastTemplate-alapú honlapon mindezeket a változtatásokat kevesebb, mint egy perc alatt elvégezhetjük a legfelsőbb szintű mintafájl, a main.tpl átszerkesztésével.

Bár jobb webszerkesztők is léteznek, a Netscape Composer is megteszi. A következő lépésekre lesz szükség: a Netscape Navigatorban válasszuk ki a *File/Open Page* menüpontot; gépeljük be az elérési utat vagy keressük ki a példakönyvtárunkban található main.tpl fájlt, majd a megjelenő párbeszédablakban bökjünk az *Open in Composer* gombra. A Netscape Composer egyetlen nagy ablakot fog megnyitni, ahol a szerkesztő részben mindössze a `{BODY}` szó olvasható (*3. kép*). Ez a main.tpl HTML-változata. A főnök által kért változtatások elvégzéséhez először válasszuk az *Edit/Select All* menüpontot. A `{BODY}`, beleértve a zárójelet is, most már sárgán virít. Ezután válasszuk a *Format/Align/Center* pontot, ezáltal elvégeztük a középre rendezést. Újfent jelöljünk ki mindent az *Edit/Select* All segítségével, majd válasszuk a *Format/Text Color* menüpontot, és a mintaválasztékból keressünk egy szép sötétzöldet. Bökjünk az *OK*-ra, és a `{BODY}` immár sötétzöldben pompázik.



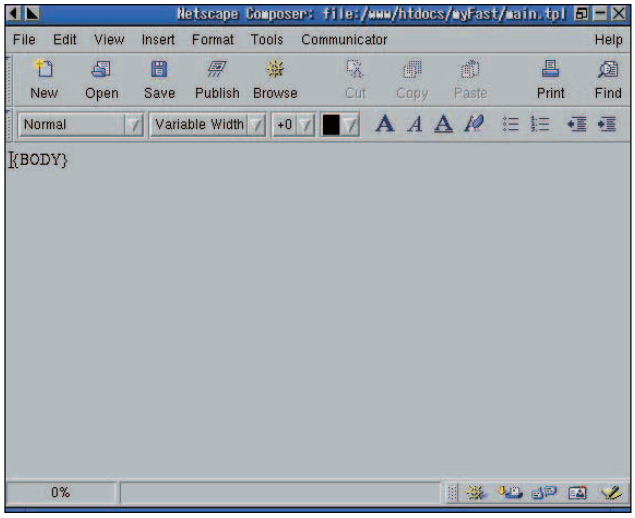
last_name	first_name	suffix	city	state	birth	death
Washington	George		Wakefield	VA	1732-02-22	1799-12-14
Adams	John		Braintree	MA	1735-10-30	1826-07-04
Jefferson	Thomas		Albemarle County	VA	1743-04-13	1826-07-04
Madison	James		Port Conway	VA	1751-03-16	1836-06-28
Monroe	James		Westmoreland County	VA	1758-04-28	1831-07-04
Adams	John Quincy		Braintree	MA	1767-07-11	1848-02-23
Jackson	Andrew		Waxhaw settlement	SC	1767-03-15	1845-06-08
Van Buren	Martin		Kinderhook	NY	1782-12-05	1862-07-24
Harrison	William H.		Berkeley	VA	1773-02-09	1841-04-04
Tyler	John		Greenway	VA	1790-03-29	1862-01-18

2. kép A próbalekérdezés eredménye

Főnökünk két kívánságát máris teljesítettük, csak egy maradt hátra. Válasszuk a *Format/Page Colors and Properties* menüpontot, ahol kattintsunk a *Use Custom Colors* gombra a *Colors and Background* fül tetején. A *Background*-ot választva a megjelenő választéktáblából keressünk egy vonzó sárgát. Kattintsunk az *OK*-ra, az

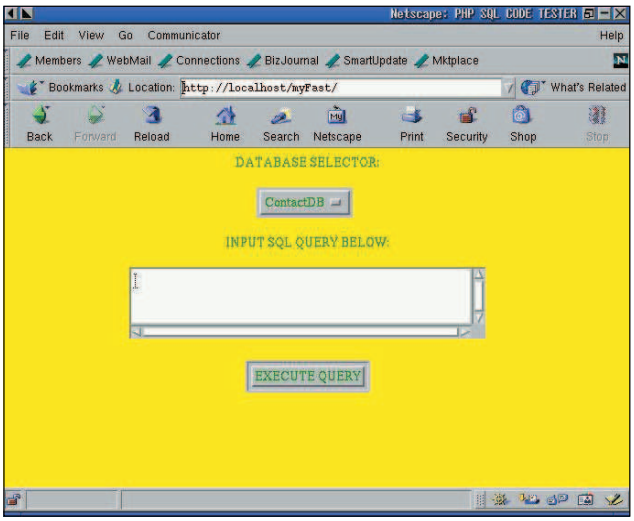
3. lista goodbye.php

```
<%/////////////// goodbye.php /////////////////

include ("class.FastTemplate.php");
$tpl = new FastTemplate(".");

$tpl->define(array(bye_message =>
                   ➥"bye_message.tpl",
                      main => "main.tpl"));

$tpl->assign(array(BYE_MESSAGE =>
        ➥"Thank you and please come again!"));

$tpl->parse (BODY, "bye_message");
$tpl->parse (MAIN, "main");
$tpl->FastPrint();

exit;
%>

Templates enable the physical separation of PHP
code and HTML. Changes to the PHP code in no
way affect the HTML, and vice versa.
```

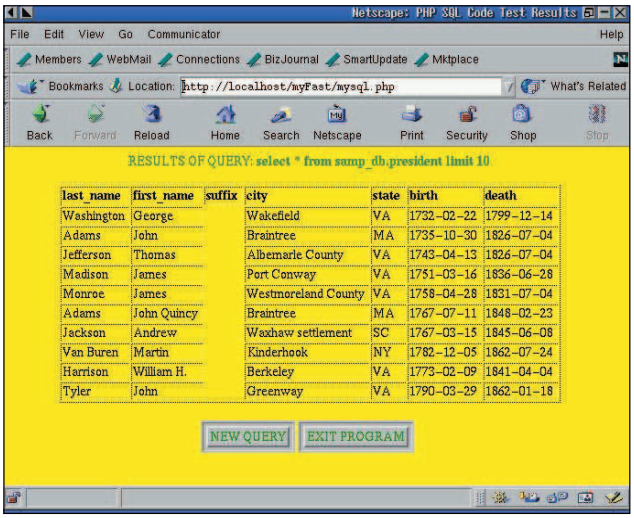


3. kép A fő sablonmodul szerkesztés közben

Apply-ra és újfent az OK-ra. A Composerben most már sárga a háttér. Ezután válasszuk a *File/Save*, és végül a *File/Close* menüpontokat.
Ha most újratöltjük a példa lekérdezőlapját, láthatjuk, hogy immár mindhárom változtatás érvényes: a háttér sárga, a szöveg zöld és minden középre rendezetten jelenik meg, ugyanakkor az alkalmazás programszerűen működik, mint eddig (lásd a *4.* és *5. képen*). Bár a példa igen egyszerű, azt hiszem, hatékonysága mindenki számára nyilvánvalóvá vált.
A FastTemplate-tel kapcsolatban néhány dolgot érdemes még megemlíteni. Először is, rengeteg apró fájllal dolgozunk, ezeket tudni kell kezelni, és a lap minden egyes elérésénél be kell olvasni a lemezről. Ez megterhelt webhelyen jelentős teljesítménycsökkenést okozhat, viszont nem túl gyakori elérésű helyeken lényegtelen szempont.
Az is elképzelhető, hogy minden egyes laphoz saját egyedi mintafájlt készít valaki, ezáltal is csökkentve a mintafájlok számát.



4. kép Az új sablon önműködően hat minden oldalra



last_name	first_name	suffix	city	state	birth	death
Washington	George		Wakefield	VA	1732-02-22	1799-12-14
Adams	John		Braintree	MA	1735-10-30	1826-07-04
Jefferson	Thomas		Albemarle County	VA	1743-04-13	1826-07-04
Madison	James		Port Conway	VA	1751-03-16	1836-06-28
Monroe	James		Westmoreland County	VA	1758-04-28	1831-07-04
Adams	John Quincy		Braintree	MA	1767-07-11	1848-02-23
Jackson	Andrew		Waxhaw settlement	SC	1767-03-15	1845-06-08
Van Buren	Martin		Kinderhook	NY	1782-12-05	1862-07-24
Harrison	William H.		Berkeley	VA	1773-02-09	1841-04-04
Tyler	John		Greenway	VA	1790-03-29	1862-01-18

5. kép A sablon hatása a lekérdezés kinézetére

Benjamin Kahn olyan weblapot tart fenn, amely a FastTemplate-hez való Cached FastTemplate nevű kiegészítéssel foglalkozik ➲ http://zoned.net:81/~xkahn/php/fasttemplate/. A Cached FastTemplate néhány további teljesítménynövelő lehetőséggel egészíti ki az eredeti csomagot. Ezzel a csomaggal a lapok egyes részeit a memóriában lehet tárolni, sőt ezek a gyorstárazó szabályok be is állíthatók. Ez mindenképpen megér egy pillantást, főleg ha már megkedveltük a FastTemplate hatékonyságát.
Ma már nem elegendő önmagában jó honlapot készíteni, nélkülözhetetlen, hogy méretezhető és karbantartható legyen. A DreamWeaver-szerű HTML-készítők használata ugyan igen könnyű, de az általuk készített HTML meglehetősen szegényes és csúnyácska. Ezekben a kesze-kusza HTML fájlokban senki sem szeret PHP-kódok után bogarászni egy egyszerű módosítás végrehajtásához. A HTML-minták használata tehát nagymértékben megkönnyítheti a webfejlesztők életét.

Bill Cunningham
nemrégiben vonult vissza az US Marine Corps-tól, ahol Solaris rendszergazdaként dolgozott. Jelenleg Észak-Karolinában Linux/MySQL-alapú honlapok webfejlesztőjeként a charlotte-i Heafner Tire Group alkalmazásában áll.