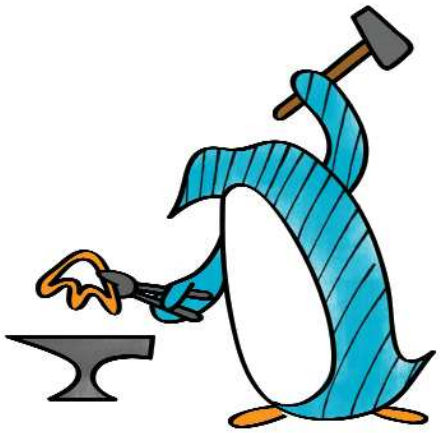


Bricolage-sablonok

A tartalomkezelő rendszer használata nem feltétlenül jár együtt az egy kaptafára készült oldalakkal. Készítsünk saját sablont!

Az elmúlt néhány hónapban a PostgreSQL-, `mod_perl`- és Apache-alapokra építkező, nyílt forrású tartalomkezelő rendszer (CMS), a Bricolage rejtelmeit boncolgattuk. A Bricolage elég nagy figyelmet kapott az elmúlt években, részben nyílt forrású engedélye, részben az alapját képező nyílt forrású technológiák, részben pedig a kereskedelmi CMS-csomagokkal összemérhető képességkészlete miatt, nem beszélve a nyílt forrású CMS lehetőségeiről.
Ebben a hónapban Bricolage-körutunk utolsó állomásaként a sablonok alkalmazási lehetőségeit ismerhetjük meg. Eddig a Bricolage számos felügyeleti és szerkesztő képességét mutattuk be, de egy CMS csak akkor igazán használható, ha a kimenete testreszabható és vonzóvá tudja tenni az eredményül kapott honlapot.

A sablonok elmélete

HTML-sablonok már elég régen léteznek, ügyes kompromisszumként valahol a statikus oldalak és a HTML programba pakolása között, ahol az a tervező számára elérhetetlenné válik. Amennyiben dinamikusan változó HTML-lapot szeretnénk, nagyon jó választás a sablonok használata.
Természetesen ezzel együtt felvetődik egy fontos kérdés: melyik sablont használjuk? Százszámra léteznek sablonozó rendszerek, ideértve a több tucatnyi Perl nyelven készültet is. Néhány ezek közül, mint a `Text::Template`, önmagában nem is HTML-sablonozó rendszer, mégis nagy sikerrel alkalmazták különféle webalapú feladatokra.
A „Melyik sablont használjuk?” vita talán csak a Linux/BSD és az Emacs/vi elsőségről szóló vitákkal mérhető össze. Szerencsére a Bricolage minden vita fölé emelkedik, ahol békében él együtt a `HTML::Mason` és a Template Toolkit (sablonozó eszközrendszer). Elméletben akár másik sablonozó rendszert is használhatunk, ez a kettő azonban olyan hatékony és népszerű, hogy a legtöbb Bricolage-felhasználót bőségesen kielégíti. Személy szerint jobban kedvelem a `HTML::Mason` rendszerét, ezért a Bricolage sablonkezelését is a Masonön keresztül mutatom be, de ha valaki esetleg Template Toolkit-rajongó lenne, nyugodtan használja azt.
A sablon tulajdonképpen a HTML-oldal tartalmának a vázlata. Minden ide tartozik, amit statikusnak szánunk, a változókat kivéve, amelyek értékét futásidőben adjuk majd meg (interpoláljuk, ahogy programozónyelven mondják). Példaképpen vegyük a következő sablont:

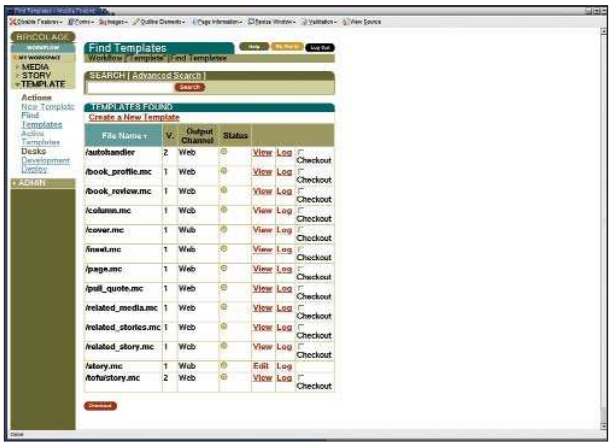
```
<html>
    <head>
        <title><% $title %></title>
    </head>

    <body>
        <h1><% $headline %></h1>
        <p><% $body_text %></p>
    </body>
</html>
```

A fenti, Mason-szabályok szerint készített sablonban mindig van cím (title), címsor (headline) és bekezdés; a `title`, `headline` és `body text` pedig változó.
A Mason két globális változót is rendelkezésünkre bocsát: a `$r`-t, ez a szabványos `mod_perl` objektum Apache kiszolgálónk belső értékeihez enged hozzáférést a Perl API-n keresztül; valamint a `$m`-et, ez az objektum a teljes Mason-környezettel és az adott Mason-sablonnal kapcsolatos további adatokat tartalmaz.
A Bricolage ehhez a keverékhez további három változót ad hozzá. A legfontosabb közülük a `$story`, amely a pillanatnyi történettel kapcsolatos adatokat tárolja. A történetek különféle elemeket tartalmaznak, amelyek maguk is további elemeket foglalhatnak magukba; az éppen pillanatnyi elemet a `$element` változón keresztül érhetjük el. Végül a Bricolage az égető (burner) nevű megoldáson keresztül küldi az oldalakat a kimenetei csatornára (ami általában, de nem szükségszerűen, a weboldal lesz).
Mielőtt folytatnánk, érdemes megismernünk a Mason önműködő kezelőit (autohandler): ezek segítségével egyedi kinézetet kölcsönözhetünk egy oldalnak. Amennyiben egy adott könyvtárhoz létezik autohandler, a fájl vagy könyvtár helyett először mindig ez hívódik meg. Ennek megfelelően, ha a */abc/def.html* és a */abc/autohandler* egyaránt létezik, a */abc/def.html* helyett a */abc/autohandler* fog elindulni. Ez elsőre egy kicsit furcsának tűnhet, és valóban az is lenne, ha nem vennénk figyelembe, hogy az autohandler az `$m->call_next()` hívás segítségével az eredeti sablont bármikor meghívhatja.

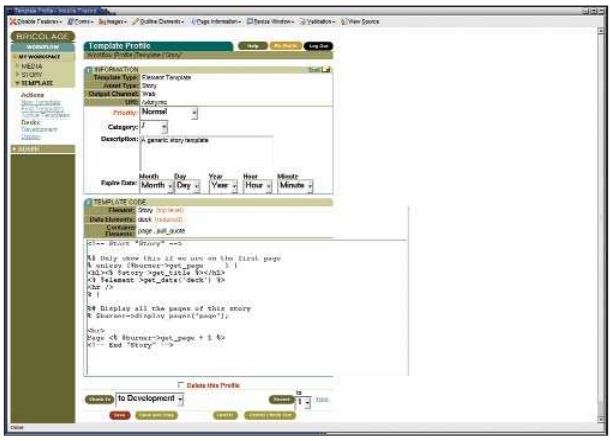
Általánosan használt megoldás, hogy az autohandler belsejébe annyi közös részt préselünk, amennyit csak tudunk, ideértve a menüket, a képeket és a fejléceket. Az autohandler pontosan olyan Mason-sablon, mint a többi, meghívásának különleges körülményeitől eltekintve. Az autohandler belsejében különféle címsorok, képek és menük közt valahová be kell szúrnunk a `$m->call_next()` hívást, amely aztán beilleszti magát a kérelmezett lapot. Ezzel megtartjuk a több sablon használatán alapuló moduláris szerkezet előnyeit, mindeközben egyetlen állomány szerkesztésével lapunkat újratervezhetjük.
Az önműködő kezelők (autohandlers) egymásba ágyazhatók, azaz a Mason az összes fellelhető autohandlert az összes szülőkönyvtárban meg fogja hívni. Tehát a */abc/def.html* hívásakor a Mason először a */autohandler* állományt hívja meg, majd a */abc/autohandler*-t, amit végül a */abc/def.html* követ. Ezáltal szakaszérzékeny (section-specific) kinézetet, illetve szakaszérzékeny (section-based) menüket, alkotórészeket hozhatunk létre.

Sablonok módosítása

Feltételezem, hogy már túl vagyunk az első történet létrehozásán és kiadásán. Amennyiben valaki nem igazán tudja, hogyan kellene történetet létrehozni és kiadni, kukkantson bele az utóbbi néhány számban megjelent Bricolage témájú írásaimba. A történet a kiadását követően valamelyik kimeneti csatornára kerül a fájl helyi másolásával vagy FTP segítségével történő távoli rendszerre juttatásával.
Történetünk kinézetét a sablon határozza meg, de még mielőtt megalkotnánk az első egyszerű sablonunkat, vessünk egy pillantást a rendszerrel együtt érkezett alapsablonokra! Lépjünk a ***Find Template*** hivatkozásra a ***Template*** menüben, a Bricolage rendszergazdai felületének bal oldalán (felhasználói név ***administrator***, a jelszó `change me now!`), majd kattintsunk a ***Search*** gombra. A szövegmezőbe ne írjunk semmit. Most a sablonok listáját kell látnunk, a rendszerben megtalálható minden elemtípushoz egyet-egyet (*1. kép*).
A Bricolage-sablonok több szempontból is hasonlítanak a történetekre: különféle asztalokon hozzuk létre, szerkesztjük és telepítjük őket; elérésük néhány felhasználóra és csoportra korlátozódik; a Bricolage pedig egyszerű változatkezelő rendszere segítségével nyomon követi a változásokat (és megakadályozza az ütközéseket). És valóban, ha megnézzük, az 1. képen azt láthatjuk, hogy valamennyi sablonhoz tartozik egy változatszám. Mindegyiket kikérhetjük a változatkezelő rendszerből, ha rákattintunk a megfelelő jelölőnégyzetre, majd a lap alján található ***Checkout*** (kikérés) gombra. A kikért sablonok az ***Active Templates*** menü ***Templates*** hivatkozás pontja alól érhetőek el.
Bricolage alatt a történet csak egy a terjeszthető elemek közül, továbbá minden elem tetszőleges számú további elemet tartalmazhat. A Bricolage néhány előre meghatározott felső szintű elemet tartalmaz, ilyen például a történet, a könyvkritika és a rovat. Ezenkívül néhány további elemet is, amelyeket más elemekbe ágyazhatunk, ilyen például az idézőjelezés (pull quote).
Ha magunk elé képzelünk egy napilapot, láthatjuk, hogy minden rovat más stílusú, még a hasonló elemeik is. Például a New York Times Metro rovatának hasábjai egészen másképpen festenek, mint a Business rovatéi, és egyik sem hasonlít az Op-ed oldalra (ezeket szaknyelven önálló rovatoknak nevezik). A Bricolage ezt a feladatot úgy oldja meg, hogy az elemeket kategóriába sorolhatjuk. Ha tehát a Sports rovatba írunk, megjelölhetjük, hogy az a ***sports*** kategóriába tartozik. Amikor a Bricolage kihelyezi rovatunkat a webre, a */sports/column.mc* sablon szerint fogja megjeleníteni. Amennyiben létezik, a Bricolage ezt a sablont használja. Ha nem létezik ilyen, kikeresi a legfelső (root) kategória ***column.mc*** sablonját. Más szavakkal, amennyiben egy elemhez felső szintű sablont rendeltünk, azt helyettesítőként vagy alapértelmezettként használhatjuk a rendszer valamennyi ilyen típusú eleméhez. A kategóriafüggő sablonok alkalmazásával lapunk valamennyi szakaszához egyedi kinézetet rendelhetünk.
Mint azt a *2. képen* láthatjuk, kikértem a */story.mc* sablont, amely a történetekhez tartozó felső szintű sablon. Megtekintés (***view***) hivatkozás helyett most már a szerkesztés (***edit***) hivatkozás is elérhető, tehát átszerkeszthetem a



1. kép Válasszuk ki a szerkesztendő elemet a Find Templates lapon!



2. kép Kikért (Checked-Out) sablon szerkesztése

sablont. Olyan módon is megtehetném ezt, hogy belépnék az ***active templates*** oldalra, amelyen hasonló szerkesztőhivatkozást találnék. Nyissuk meg a sablont szerkesztésre, így a 2. képhez hasonló eredményt kell kapunk.
A sablonok szerkesztése igen hasonló a történetek vagy más elemtípusok szerkesztéséhez, eltekintve attól, hogy most épp azt a tárolót módosítjuk, ahol a történetünket később majd elhelyezzük. Amennyiben csak statikus HTML-szerkezeteket

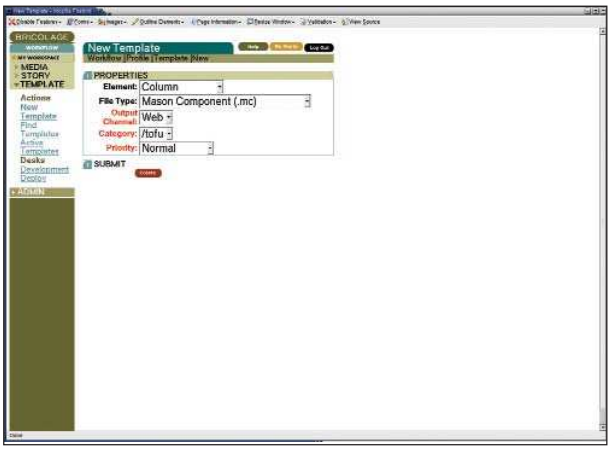
használunk, minden elem pontosan ugyanúgy fog kinézni. Ezért a trükk éppen az, hogy az oldal dinamikus tartalmának létrehozásához felhasználjuk az előre meghatározott `$story`, `$element` és `$burner` objektumokat. Példaképpen álljon itt az alapértelmezett */story.mc* sablon:

```
<!-- Start "Story" -->

%# Only show this if we are on the first page
% unless ($burner->get_page     ) {
<h1><% $story->get_title %></h1>
<% $element->get_data('deck') %>
<hr />
% }

%# Display all the pages of this story
% $burner->display_pages('page');

<br>
Page <% $burner->get_page + 1 %>
<!-- End "Story" -->
```

Láthatjuk, hogy a fenti sablon meglehetősen egyszerű. Egy komolyabb webhely ide különféle elemeket illeszthet be: CSS stílusokat vagy egyéb, az oldalt azonosító statikus szövegeket. A */story.mc* alapváltozata a következőket végzi el:

- A `$burner->get_page()` változóból lekéri a pillanatnyi oldalszámot. A lapszámozás 0-val kezdődik, mi azonban az első oldalon a címet és az elemek listáját jelenítjük meg. A címet a `$story` változóból vesszük a `$story->get_title()` kifejezés segítségével, az adatokat pedig magából az elemből szedjük ki. Figyeljük meg az `$element->get_data()` általános alakú hivatkozást – az elem bármely mezőjéhez hozzáférhetünk az `$element->` meghatározás alkalmazásával.
- A `$burner->display_pages('page')` meghívásával az égetőből kikérve megjelenítjük a történetet.
- Végül ismét felhasználjuk a `$burner->get_page()` utasítást, hogy az oldal alján megjeleníthessük a lapszámot.

Mi történik, ha eltávolítjuk a lapszámot és a helyére a sablon tetején a saját statikus HTML-szakaszunkat szúrjuk be? Változtatásaink a rendszer valamennyi történetén meg fognak látszani. Nem szabad elfelejtenünk, hogy nem minden elem történet, ennek megfelelően a */story.mc* megváltozása nincs hatással a rovatokra, a könyvkritikákra és más elemtípusokra.
Miután elkészültünk a */story.mc* szerkesztésével, rákattinthatunk az oldal alján található ***Check-In*** gombra. Ezzel beolvastuk a sablont, így most már elküldhetjük az (alapértelmezés szerinti) fejlesztőisablon-asztalra (development template desk) vagy azonnal telepíthetjük. Ez az utóbbi lehetőség különösen akkor hasznos, ha a sablonon felfedezünk valamilyen hibát, ami az egész lapra hatna; könnyen módosíthatjuk a sablont, telepíthetjük és azonnal megnézhetjük az eredményt.
Végül megfigyelhetjük, hogy a */story.mc* egyáltalán nem tartalmaz <html> vagy <title> tagokat. Az ok egyszerű, ezeket a részeket az autohandler foglalja magában.
A */autohandler* sablon, amelyet a ***Template*** menüponton



3. kép Kimeneti csatorna és kategória kiválasztása az új sablonunkhoz

keresztül megtekinthetünk, kikérhetünk és szerkeszthetünk, alapértelmezés szerint a következőképpen néz ki:

```
<!-- Start "autohandler" -->
<html>
    <head>
        <title><% $story->get_title %></title>
    </head>
    <body>

    % $burner->chain_next;
    </body>
</html>
<!-- End "autohandler" -->
```

A teljes oldalrendszeren keresztül érvényes autohandler a történet címét a megfelelő HTML <title> tagok közé helyezi, s a `$burner->chain_next()` meghívásával egyúttal a megfelelő laptartalmat is beilleszti.
Ha például általános CSS stíluslapot szeretnénk használni vagy közös menüt akarunk minden oldal tetejére szerkeszteni, esetleg cégünk jelvényét kívánjuk az összes lap tetején megjeleníteni, ebben az autohandlerben tehetjük meg a legkönnyebben. Minthogy az önműködő kezelők egymásba ágyazhatók, a teljes oldalrendszerhez lehet egy általános autohandlerünk, valamint szakaszra jellemző kezelőink az egyes szakaszokhoz.

Sablonok készítése

Mostanáig létező sablonokkal foglalkoztunk, mindazonáltal új sablont létrehozni is éppen ilyen könnyű feladat. Egyszerűen lépjünk a ***Template*** menüpontra és kattintsunk a ***New Template*** hivatkozásra; ott a *3. képhez* hasonlókat kell látnunk. Meg kell adnunk a sablonunkhoz tartozó kimeneti csatornát és a kategóriát. Kattintsunk a ***Next***-re, majd megadhatjuk, hogy melyik kategóriára vonatkozzon a sablonunk.
A kategória-csatorna-elem együttesnek egyedinek kell lennie, ennek megfelelően egy csatornához, kategóriához vagy elemhez több sablont is rendelhetünk. De a ***Web output channel in the root (/)*** kategóriájának ***story*** eleméhez csak egyetlen sablon tartozhat. Ha megpróbáljuk megszegni ezt a szabályt, a Bricolage figyelmeztet bennünket,

hogy erre a kombinációra már létezik sablon. E nehézségre több megoldás is létezik; az egyik, hogy egyszerűen új elemtípust hozunk létre, a másik, hogy új kategóriát készítünk, végül akár az adott kombinációhoz tartozó eredeti sablont is megváltoztathatjuk. A legjobb megoldás mindig az adott elérendő céltól függ.
Most készítsünk új sablont a *tofu* kategória rovataihoz, amelyet Bricolage alatt a */tofu/column.mc* állomány képvisel. Miután rákattintok a *Create* gombra, egy szerkesztőoldalt kapok, ahol létrehozhatom vagy módosíthatom a sablonomat. Én a sablonomat rendkívül egyszerűre tervezem:

```
<!-- Start "tofu/column" -->

%# Display this story
% $burner->display_pages(´page´);

<!-- End "tofu/column" -->
```

Figyeljük meg, hogy a meghatározás köré HTML-megjegyzéseket helyeztünk. Ezáltal könnyebb lesz a sablonjainkban megkeresni a hibát, miután az HTML alakban a felhasználó böngészőjére került. A saját tapasztalataim alapján bátran kijelenthetem, hogy a Mason egymásba ágyazott sablonjai, különösen többszörös önműködő kezelőhasználattal, igencsak őrjítőek tudnak lenni.
Miután a *Check-In* menüből kiválasztom a telepítést (deploy), majd rákattintottam a *Check-In* gombra, sablonom beépült a rendszerbe. Ettől kezdve bármely rovat, amely a *tofu* kategóriába tartozik, az általános sablon helyett az új sablonunkat használja.
Természetesen, ha vissza szeretnék lépni, hogy átszerkesszük a sablont, a korábban látott módon nyugodtan megtehetjük – kikeressük, kikérjük, majd átszerkesztjük.

Összegzés

A Bricolage, mint minden komoly CMS, sablonok segítségével teszi egyszerűvé az egységes oldalkép kialakítását. Minthogy a Bricolage olyan általános nyílt forrású eszközökön alapul, mint a `mod_perl` és az Apache, kihasználhatja a `mod_perl`-ben már létező sablonozás előnyeit, ideértve a `HTML::Mason` és a Template Toolkit-megoldásokat. Ebben a hónapban megnéztük, hogyan hozhatunk létre és módosíthatunk különféle elemtípusokhoz és kategóriákhoz tartozó sablonokat, ilyen módon kötöttségek nélkül juthatunk hozzá a honlapunk kinézetének egységesítését biztosító rugalmassághoz.
A cikkhez tartozó Kapcsolódó címek megtalálhatóak a ➲ http://melleklet/linuxvilag.hu/Bricolage könyvtárában.

Linux Journal 2004. március, 119. szám

Reuven M. Lerner (➲ http://www.lerner.co.il/atf) Nyílt forrású programokra, valamint web- és adatbázis-alkalmazásokra szakosodott tanácsadó. Könyve, a Core Perl, 2002 januárjában jelent meg a Prentice Hall gondozásában. Reuven feleségével és lányaival nemrég költözött Chicagóba.