Zope lapsablonok

Fedezzük fel, hogyan készíthetünk hasznos és együttműködő dinamikus weblapokat ZPT-tulajdonságok és HTML együttes felhasználásával.

A web hőskorában csaknem minden weblapkészítő egyben valamilyen programozó volt. Biztosak lehetünk benne, hogy minden webmester saját maga telepítette a webkiszolgálóját, tudta, hogyan kell kézzel HTML-t készíteni, és a legfontosabb CGI-parancsfájlokat is meg tudta írni magának.

Idővel azonban számos szerkesztő, tervező, illetve más nem programozó szakember is részesévé vált a webfejlesztés folyamatának. Bár gyakran megvolt a lehetőség, hogy őket is megtanítsák a HTML nyelv használatára, a HTML csak statikus weboldalak készítéséhez volt elegendő.

A dinamikus oldalak azonban, amelyek akkoriban szinte kivétel nélkül CGI-parancsfájlokon keresztül készültek, egészen másképp működnek. Ha ugyanis a tervező meg szeretné változtatni egy statikus lap háttérszínét, egyszerűen csak módosítja a megfelelő HTML-fájlokat (vagy manapság egy globális stíluslapot), és már készen is van. Ha ez a tulajdonság azonban egy CGI-programban rejtezik, a tervezőnek a programozótól kell kérnie a lap megváltoztatását. Ez a megoldás senkinek sem jó: a tervező nem tud új ötleteket próbálgatni, a programozó pedig apró és idegesítő változtatásokat kénytelen végezni, miközben a többi programozói munka szünetel.

Néhány éve erre a gondra a legelterjedtebb megoldás az úgynevezett sablonok használata, amelyek valójában a HTML és programozási nyelv keverékei. A talán legismertebb üzleti példa az ilyen sablonokra a Microsoft Active Server Pages (ASP) nyelve, de a Sun JavaServer Pages (JSPs) szintén igen népszerű. A nyílt forrású programok fejlesztői számos saját, magas minőségű sablonmegvalósítást készítettek, ideértve a HTML::Masont (amely `mod_perl`-lel működik a legjobban), a PHP-t (kifejezetten websablonokhoz készült nyelv) és ADP-t (az AOLServer-hez hozták létre).

Az ilyen sablonok mögött meghúzódó alapelképzelés igen egyszerű: alapértelmezés szerint minden statikus HTML, kivéve, ami valamilyen különleges zárójelek közé kerül. Amikor a tervező ilyen sablonokkal dolgozik, általában csak annyit kell mondani neki: „Mindent módosíthatsz, ami nincs `<%` és `%>` jelek között.” Tulajdonképpen ez a legtöbb esetben igen jól működik.

Csakhogy idővel az ilyen típusú sablonok hátrányai is egyre inkább nyilvánvalóvá váltak. Például mi történik, ha egy olyan ciklust akarunk írni, ami adatbázisból lekért elemeken megy végig, de tartalmuktól függően minden elemet más és más háttérszínnel szeretnénk megjeleníteni? Ebben az esetben nem mondhatjuk meg a tervezőnek, hogy hagyja békén a kódot, hiszen a kód és a HTML igencsak összefonódik.

Az utóbbi néhány hónapban vizsgált Zope névre hallgató alkalmazáskiszolgáló ezt a gondot az általuk DTML-nek (Dynamic Template Markup Language) nevezett nyelvvel próbálta áthidalni. Mint azt néhány hónapja láthattuk, a DTML egy HTML-szerű szerkezettel bíró nyelv, amely a tervezőknek és a fejlesztőknek egyaránt lehetővé teszi a weblapok kialakítását. A DTML hatékony, rugalmas és könnyen megérthető (legalábbis akkor, ha valaki programozó). Amennyiben valaki nem programozó, előfordulhat, hogy a DTML is kissé nehezen érthető a számára, különösen, ha eredményeket kér le egy adatbázisból (mint azt a múlt hónapban láthattuk). Továbbá a HTML-szerkesztők nemigen tudják, hogy mihez kezdjenek a DTML-tagokkal, ami könnyen vezethet a DTML-lapok akaratlan összezavarásához.

Ezen okok miatt a Zope Corporation (korábban Digital Creations), amely a nyílt forráskódú Zope alkalmazáskiszolgálót fejleszti, most arra ösztönzi a fejlesztőket, hogy próbálják ki a Zope Page Templates (ZPT, azaz Zope lapsablonok) elnevezésű, új sablonmegközelítést, amely képes megoldani a gondok nagy részét. Ebben a hónapban a ZPT-t fogjuk megvizsgálni, végigvéve, hogy milyen lehetőségeket kínál a dinamikus oldalak készítésére.

Mi is a ZPT?

A ZPT három dolgot használ ki:

- a HTML XML formátumban is leírható, amit gyakran XHTML-ként emlegetnek;
- az XML lehetővé teszi, hogy elkülönített névterek segítségével a különböző dokumentummeghatározások tagjait és tulajdonságait összekeverjük;
- a WYSIWYG HTML-szerkesztők az általuk fel nem ismert tulajdonságokat általában figyelmen kívül hagyják (de azért megőrzik).

A ZPT így magát a HTML-meghatározást módosítja azáltal, hogy más névtérben elhelyezkedő tulajdonságokat ad hozzá. Mivel ezek a tulajdonságok egy másik névtérben találhatók, XML és XHTML alatt teljesen törvényszerűek. Minthogy tulajdonságokról és nem tagokról van szó, a legtöbb HTML-szerkesztő figyelmen kívül hagyja őket, és a tagot úgy jeleníti meg, mintha a tulajdonság nem is létezne. Mivel a böngészők figyelmen kívül hagyják azokat a tulajdonságokat, amelyeket nem ismernek fel, a ZPT-lapokat akkor is meg tudják jeleníteni, ha azok előtte nem mennek át a Zope-on. Más szavakkal: a ZPT lehetővé teszi, hogy a programozó egy olyan lapot építsen fel, amit a tervező bármilyen neki tetsző eszközzel megnézhet és módosíthat. Természetesen a sablon dinamikus elemei csak akkor kelnek életre, ha a lapot Zope-on keresztül nézzük.

A HTML-ben minden tulajdonságnak van egy neve és egy értéke, például a

```
<a href="http://www.lerner.co.il/"
➥>Reuven oldala</a>
```

esetében a `href` tulajdonság értéke:

`http://www.lerner.co.il/`

A nevet és az értéket egyenlőségjel (=) választja el, ahol az érték idézőjelek közé kerül. Mindez semmit sem változik a ZPT-vel való munka során, eltekintve attól, hogy a tulajdonság neve TAL (Template Attribute Language, vagyis sablontulajdonság-nyelv) formában van megadva. A TAL több különböző lehetséges tulajdonságnevet meghatároz, amelyek mindegyike megjelenítéskor a sablon valamilyen módon történő megváltoztatására utasítja a Zope-ot.
A tulajdonságneveket a TAL határozza meg, de mi adja meg a tulajdonságértékeket? Ehhez a TALES-t (TAL Expression Syntax, azaz TAL kifejezésszabályok) használjuk. A TALES az adatok számára számos forrást határoz meg, ideértve a Python-kifejezéseket és a környező Python-névtér értékeit. A TAL-ból származó tulajdonságnév (amely azt mondja meg a ZOPE-nak, hogy a környező tagot hogyan kezelje) és a TALES tulajdonságérték (amely azt mutatja meg a ZOPE-nak, hogy a TAL kifejezésben milyen értéket használjon) együttese lenyűgöző rugalmasságot ad a sablonkészítéshez. Ráadásul azáltal, hogy a TAL és a TALES közzétett és nyílt forrású formátum, bármikor bővíthetjük, ha esetleg még meg nem valósított, különleges igényünk támadna.

Néhány egyszerű példa

Most hogy már ismerjük a ZPT mögött rejtező elméletet, ideje megismerkednünk a gyakorlati alkalmazással. Első példánk az a vázdokumentum lesz, amit a Zope új sablon létrehozásakor készít nekünk. Létrehozásához lépjünk a Zope-kiszolgáló */**manage*** címére, és a jobb felső sarokban található ***Add product*** (termékfelvétel) listából a ***page-template*** (lapsablon) elemet válasszuk (ha semmiféle „page template”-et nem látunk a menüben, a ZPT-termék valószínűleg nincs feltelepítve – ilyenkor Zope-változatunkat frissítenünk kell egy újabbra, hogy használhassuk a ZPT-t). Mint azt már megszokhattuk a Zope világában, ezután új termékünkhöz egy ID-t kell rendelnünk (tehát meg kell adnunk egy olyan egyedi karaktersorozatot, amely a címben jelenik meg). Gépeljünk be valamilyen rövid nevet, majd kattintsunk az ***Add and edit*** pontra.
Ha a fenti utasításokat egy DTML-dokumentum vagy tagfüggvény esetében követnénk, egy szerkesztőablakhoz jutnánk, amely a DTML vázát tartalmazza. Ugyanez a lapsablonok esetében is igaz, eltekintve attól, hogy a vázlat most egyértelműen TAL- és TALES-kifejezéseket tartalmaz:

```
<html>
  <head>
     <title tal:content="template/title">
      A c m</title>
  </head>
  <body>

     <h2><span tal:replace="here/title_or_id">
          Tartalom c m vagy azonos t </span>
        <span tal:condition="template/title"
             ➥tal:replace="template/title">
           elhagyhat  sablon id
        </span></h2>

     Ez itt egy LapSablon
     <em tal:content="template/id">sablon
     ➥id</em>.
  </body>
</html>
```

A dokumentum lehet, hogy rövidnek tűnik, de hatékonyan mutatja be a TAL és TALES rendszert, illetve azt, hogy miképpen használhatjuk őket ténylegesen egy dokumentumban.
A dokumentum a következő TAL-tulajdonságokat használja:

- A `tal:content` tulajdonság: a tag tartalmát a TALES-kifejezés értékével helyettesíti. Így a fenti példában például az „A cím” szavak a `template/title` TALES-kifejezés értékével fognak helyettesítődni, erről azonban még bővebben szót ejtünk. A `<title>` és `</title>` tagok tulajdonságaikkal együtt változatlanok maradnak, de a két tag közötti szöveg meg fog változni, amennyiben a Zope jeleníti meg a sablont.
- A `tal:replace` tulajdonság: nagyon hasonló a `tal:content`-hez, de a tartalmat és az azt körbezáró tagot is helyettesíti. Gyakran használják a `<span>` taggal, amely voltaképpen az ilyen esetek kezelésére szánt helyfoglaló tag. Tehát az első `<span>` tagtól az utolsó `</span>` tagig, ideértve a magukat a tagokat, minden a `here/title_or_id` TALES-kifejezéssel helyettesítődik.
- A `tal:condition` tulajdonság: csak akkor jeleníti meg a tartalmát, ha az értéke igaz, azaz nem nulla, nem üres karaktersorozat vagy üres lista, és nem a ZPT beépített `nothing` változója.

Észrevehettük, hogy a második `<span>` tag két TAL-tulajdonságot is tartalmaz: a `tal:condition`-t és a `tal:replace`-t. Amikor a Zope egy adott tagon belül több TAL-tulajdonsággal találkozik, először a meghatározásokat, azután a feltételeket, majd az ismétlő ciklusokat, végül pedig a `content` és `replace` részeket hajtja végre (nem adhatunk meg `content` és `replace` tulajdonságokat egyazon tagon belül, hiszen logikailag kizárják egymást). A második `<span>` tagban a Zope elhagyható sablonazonosítót ad meg, amennyiben a `template/title` TALES-kifejezés igaz.
A `content`, `replace` és `condition` tulajdonságokon kívül a TAL további három tulajdonságot is meghatároz:

- `tal:repeat` lehetővé teszi, hogy végiglépkedjünk egy lista elemein. Ha sablonunk egy keresés eredményét, adatbázissorokat, esetleg egy könyvtár fájljait jeleníti meg, könnyedén végiglépkedhetünk a tartalmon, több különböző módon jelenítve azt meg.
- `tal:attributes` lehetővé teszi, hogy helyettesítsük (vagy kibővítsük) a befoglaló tag tulajdonságait. Például futásidőben is beállíthatjuk az `anchor` (`<a>`) tag `href` tulajdonságát, ha az adott `<a>` tagban a `tal:attributes` tulajdonságot használjuk. A tulajdonságot a lap futásidejű kiértékelése során a TALES-kifejezés pillanatnyi értéke fogja meghatározni.
- `tal:define` lehetővé teszi, hogy új változókat állítsunk be, amelyek azután a befoglaló tag belsejéből elérhetők lesznek. Ez változóérték másik TAL-tagokba kerülhet vagy megjeleníthetjük.

TALES

A TAL-t kitárgyaltuk, jöjjenek a TAL-tulajdonságokhoz rendelhető TALES-kifejezések! A TALES-kifejezések legegyszerűbb fajtája a „path expression” (útvonal-kifejezés), amely egy Zope-objektumot ír le relatívan a sablonhoz, annak tárolójához vagy a lekérdezéshez képest. Az útvonal-kifejezések az URL-ekre hasonlítanak, azzal az eltéréssel, hogy a (/) gyökérobjektum helyett névvel kezdődnek. Az útvonal-kifejezés utolsó

tagja általában egy tulajdonság lesz, ami megjeleníthető (a `tal:content`-tel), kipróbálható (a `tal:condition`-tel) avagy ismételhető (a `tal:repeat`-tel).
A ZPT vázdokumentum négy TALES útvonal-kifejezést használ:

- `here/title_or_id`, amely az elhagyható `title` tulajdonság értékét (amennyiben létezik), vagy a kötelező `id` tulajdonságot (ha nincs cím megadva) adja át a `tal:replace` tulajdonságnak.
- `template/title`, amely a pillanatnyi sablon (elhagyható) címét adja át a `tal:condition`-nek, majd az azt követő `tal:replace`-nek.
- `template/id`, amely a pillanatnyi sablon `id` tulajdonságát jeleníti meg.

A `template` -e kezdődő TALES útvonal-kifejezések magára a sablonobjektumra hivatkoznak, míg a `here` szócskával kezdődők arra az objektumra értendők, amelyre a sablont alkalmazzuk. Ez lehet maga a sablon, de akár egy teljesen más objektum is. További relatív jelölések: `request` (a Zope HTTP-kérelem objektuma), `repeat` (a pillanatnyi `tal:repeat` ciklusadatok), `options` (a sablonnak átadott értékek) és `container` (a pillanatnyi objektum könyvtára).
A TALES útvonal-kifejezések nagyszerűek, néha azonban nem olyan rugalmasak, mint szükséges lenne. A TALES ezért lehetővé teszi, hogy ha a TALES-kifejezést `python:` jelöléssel kezdjük, Python-kódot adjunk vissza. Például egy egyszerű számítás eredményét a következő kóddal foglalhatjuk a sablonba:

```
<p>2 + 2 = <span tal:replace="python:2+2">
          ➥szÆm</span></p>
```

Létezik még egy `string:` TALES-előtag is, amely azt jelenti, hogy az értéket egyszerű szövegként kell értelmezni. A szöveges kifejezések tartalmazhatnak $ (dollárjellel) kezdődő helyettesítendő változóneveket is, hasonlóan a Perl- és parancsfájlokban megszokottakhoz. A változóneveket kapcsos zárójelekkel ({ és }) is körbekeríthetjük, valahogy így: ${x}.

Szabványos kinézet (Look and Feel)

Amit eddig láthattunk, az mind igen szép a dinamikus tartalom-létrehozáshoz. De a DTML (vagy bármely más kifinomult kiszolgálóoldali makrónyelv) egyik legnagyszerűbb tulajdonsága, hogy a menüket az egyik dokumentumban adjuk meg, a fejléceket egy másikban, a lábléceket pedig egy harmadikban, majd a további lapokban szükség szerint beolvassuk őket.
Az egyik megoldás, hogy három külön sablont készítünk (menu, header és footer) a pillanatnyi könyvtárban, és a következő TAL-kifejezésekkel importáljuk őket:

```
<span tal:replace="container/menu">
➥ide j n a men </span>
```

A TALES belenéz a pillanatnyi sablon tárolójába, lekéri a menüobjektumot (ami történetesen maga is lapsablon), majd a tartalmát a jelenlegi dokumentum `<span>` tagjainak helyére illeszti.
A DTML rugalmasságát makrók alkalmazásával is kihasználhatjuk. A makrók gyakran szerepelnek programozási nyelvekben, és lehetővé teszik, hogy futásidőben kiértékelhető kifejezéseket készítsünk. A ZPT makró nyelvét METAL-nak nevezik, és akárcsak a TAL vagy a TALES esetében, ezt is a HTML-tulajdonságok közt határozhatjuk meg, illetve hívjuk meg a `metal:` XML névtér alatt. A METAL makrók képesek foglalatokat (slot) vagy értékeket meghatározni, amelyekhez további értékeket rendelhetünk. Nem nehéz elképzelni, hogyan lehetne olyan makrót készíteni, amely a dokumentumokat a makró által nyújtott foglalatokba helyezve a teljes honlapkinézetet kezelni képes. A makró meghatározása megváltoztatása a teljes honlapkinézetet hatékonyan megváltoztatná.

Összefoglalás

Amikor első ízben hallottam a ZPT-ről, biztos voltam benne, hogy ez is csak egy újabb sablonkészítési eljárás, amely semmilyen más módszerrel nem cseresztabatos. Idővel azonban meggyőződhettem róla, hogy a ZPT valóban ügyes és elegáns elképzelés, mégpedig az a fajta, amely a fejlesztők és a tervezők munkáját egyaránt megkönnyíti. Bár nem helyettesítheti teljes mértékben a DTML-t, úgy érzem, DTML-kódjaim nagy részét ki tudom váltani TAL-, TALES- és METAL-kifejezésekkel. Nagyon várom már, hogy az elkövetkező hónapokban és években láthassam e módszerek további fejlődését és még jobb illeszkedését a Zope-hoz.

Linux Journal 2002. július, 99. szám

Reuven M. Lerner (reuven@lerner.co.il) kisebb, webes és internetes módszerekkel foglalkozó tanácsadó cég tulajdonosa és vezetője. A cikk megjelenésének időpontjában valószínűleg már végleg elkészült Core Perl című könyvével, melyet idén jelentet meg a Prentice-Hall. Az ATF honlapon érhető el (➲ http://www.lerner.co.il/atf/).

Kapcsolódó címek

A ➲ http://zope.org honlap szép mennyiségű ZPT-vel kapcsolatos adatot tartalmaz. A The Zope Book utóbbi kiadásai (a korábbi változat a New Riders jóvoltából nyomtatott formában is elérhető) két ZPT-ről szóló fejezetet tartalmaznak, ideértve néhány TAL-, TALES- és METAL-példát:
➲ http://www.zope.org/Members/michel/ZB/ZPT.dtml és
➲ http://www.zope.org/Members/michel/ZB/AdvZPT.dtml

DTML-ben jártas Zope-fejlesztők számára készült ZPT-bemutató írás található a következő címen:
➲ http://www.zope.org/Wikis/DevSite/Projects/ZPT/IntroductionForDTMLers

A TAL, a TALES és a METAL meghatározása rövid, olvasható és a következő helyen érhető el:
➲ http://www.zope.org/Wikis/DevSite/Projects/ZPT/TAL,
➲ http://www.zope.org/Wikis/DevSite/Projects/ZPT/TALES és
➲ http://www.zope.org/Wikis/DevSite/Projects/ZPT/METAL
A meghatározások az idők során jó néhány változáson keresztül mentek, ezért lehetőleg mindig a legfrissebb változatot olvassuk el.