CMF-típusok

Egy tartalomkezelő csak testreszabás után használható igazán. Kezdjük munkánkat egy olyannal, ami elég hatékony ahhoz, hogy éppen úgy működjön, mint a szervezetünk.

Az előző hónapokban általánosságban foglalkoztunk a tartalomkezelő rendszerekkel (CMS), majd külön kiemeltük a Zope tartalomkezelő keretrendszerét (CMF). A Zope CMF olyan eszközt ad a fejlesztők kezébe, amellyel megalkothatják a saját tartalomkezelő rendszerüket. Természetesen az, aki már dolgozott CMS-rendszerrel, tudja, hogy felhasználás előtt még a kereskedelmi változatok legtöbbjét is alaposan át kell szabni és újra kell dolgozni. A Zope nemcsak egyszerűen az alapprogram árát csökkenti, de gazdag környezetet is kínál, amelyben könnyedén fejleszthetjük és ízlésünkhöz igazíthatjuk a CMS-t.

A CMF-oldalak létrehozása során (az oldal gazdájaként) dokumentumokat adhatunk a rendszerhez, illetve módosíthatunk és törölhetünk. Kattintsunk a könyvtártartalom hivatkozásra, majd bökjünk a *New...* gombra, és válasszuk ki, hogy milyen típusú dokumentumot szeretnénk, illetve milyen azonosítóval rendelkezzen, majd kattintsunk az *Add* gombra. Gépeljük be a segédadatokat (ez a cím, a leírás, a téma és a tartalomtípus), kattintsunk a *Change&Edit* gombra, gépeljünk be valami tartalmat, és már kész is vagyunk.

Igaz ugyan, hogy a már meglévő tartalomtípusok egy-egy egyszerű oldal esetében elegendőnek bizonyulhatnak, a kifinomultabb oldalakhoz általában egyedi változatokra van szükség: a saját típusokra. A CMF pontosan ennek a lehetőségét kínálja fel. Ebben a hónapban a CMF-be illeszthető típusokkal foglalkozunk, megtudhatjuk, hogy miképpen dolgozzunk velük, viselkedését hogyan szabjuk az igényeinkhez, milyen módon telepítsünk újabb elemeket, vagy akár hogyan készítsünk saját tartalmat kezelő egyedi típusokat.

A CMF-típusok

Az új típusok létrehozásának legegyszerűbb módja, ha a CMF beépített, webalapú típuskiterjesztés rendszerét használjuk. Ennek segítségével olyan új típusokat hozhatunk létre, amelyek tagfüggvényeiket, tulajdonságaikat, műveleteiket, megjelenítő sablonjaikat és ikonjaikat megoszthatják a többi típussal. Amikor a webalapú kiterjesztésrendszerrel új típust hozunk létre, az elemek bármelyikét megváltoztathatjuk, kivéve a tagfüggvényeket és a tulajdonságokat (properties). Más szavakkal, újonnan létrehozott típusunk akár egészen más kinézettel is rendelkezhet, mint a szülőtípusa, a viselkedése azonban továbbra is nagyon hasonló lesz.

A példa kedvéért vegyük a *types* eszközként ismert elemet, amihez a CMF-oldalunk kezelőfelületének *portal_types* elemére kattintva juthatunk el. Ha még nem adtunk meg CMF-oldalt a Zope alatt, könnyedén készíthetünk egyet a webalapú Zope-kezelőfelület jobb felső sarkában található *Add...* menü *CMF Site* pontjának kiválasztásával. Miután elkészítettük az oldalt, a kezelőfelületből a hozzátartozó ikonra kattintva jó néhány, csavarkulcsra emlékeztető ikonnal ellátott különböző beállításeszközt fogunk találni.

Amikor első ízben lépünk be a *types* (típusok) eszközbe, megfigyelhetjük a jelenleg megadott CMF-típusokat, többek közt a *folders* (mappák), *documents* (dokumentumok), *news items* (hírelemek), *links* (hivatkozások) és *topics* (témák) típusait. A kiválasztott típus nevére kattintva megnézhetjük, illetve megváltoztathatjuk a típusokhoz tartozó tulajdonságokat és műveleteket. Ha például a *File* tartalomtípus működését meg szeretnénk változtatni, kattintsunk a *File*-ra. Az oldal tetején új fülkészlet jelenik meg, amelyek közül csak a *properties* (alapértelmezett tulajdonságok) és az *actions* (műveletek) lesz az, ami nem része a hagyományos Zope-készletnek. Igazság szerint a tulajdonságok fül a szabványos Zope-fülek közt is megtalálható, de a CMF-típusok számos szokatlan tulajdonságnévvel is bírnak.

A megszokott szabványtulajdonságokon túl minden típus néhány, a működését befolyásoló egyéb tulajdonsággal is rendelkezik:

- *Icon*: karaktersorozat, ami azt írja le, hogy az adott típushoz milyen ikon jelenjen meg.
- *Product metatype*: a Zope-termék metanevét adhatjuk meg itt. A metaneveket az *Add...* menüpontban a Zope-kezelőfelületén (Management Interface) használjuk. A CMF hasonló *Add...* menüjében úgyszintén ezt a nevet találjuk.
- *Product name*: azt a Zope-terméket jelöli meg, amelyikben a CMF-típust meghatároztuk. Minthogy a *File* és a *News* elemtípusok már az eredeti CMF-telepítésben is benne vannak, itt a *CMFDefault* termék neve szerepel. Ami igaz is, hiszen ha belenézünk a */lib/python/products/CMFDefault* könyvtárba, amely, CMF 1.3-as változat alatt a *CMF-1.3/CMFDefault* könyvtárra mutató közvetett hivatkozás, megtalálhatjuk a tartalomtípusokat meghatározó `File.py` és a `NewsItem.py` Python-modulokat. Ha kíváncsiak vagyunk, hogyan állíthatjuk be a tulajdonságok alapértékeit, nézzük meg a `factory_type_information` változót bármelyik CMF-típus akármelyik moduljában.
- *Product factory method*: megadja azt a tagfüggvényt, amelyet a CMF a típus új példányainak a létrehozásához hív majd meg.
- *Filter content types* és *Allowed content* types: ez a két bejegyzés ugyan két különálló tulajdonság, de együvé tartoznak. Mindkét tulajdonság valamennyi CMF-típusban megtalálható, kizárólag a mappa stílusú elemekre vonatkoznak. Ilyen például a *Folders* vagy a *Topics*. Az első, a `Filter content types` kétértékű változó, amely az `Allowed content types` érvényességét kapcsolja ki-be. A második, az `Allow content types` változó azt mutatja meg, hogy milyen altípusok kerülhetnek az adott típusba. Tehát például, ha egy olyan könyvtárat szeretnénk létrehozni, ami kizárólag hírelemeket tartalmazhat, ezt a *yes*-re való kattintással könnyedén megtehetjük, majd megmutatva, hogy mely típusokat engedélyezzük.

Új típus létrehozása

Új CMF-típusokat úgy hozhatunk létre a legkönnyebben, ha egy már létező típusra alapozva a webalapú CMF-típuskészítő eszközt használjuk. Ez a módszer ugyan nem engedi meg, hogy a típussal kapcsolatos mezőket vagy tagfüggvényeket megváltoztassuk, viszont lehetővé teszi, hogy szabadon állíthassuk be a típus műveleteivel kapcsolatos engedélyeket, megadjuk, hogy a típus megvitatható-e, sőt még az adattípus megjelenítésének a módját is megengedi.

Például lépjünk be a ***portal_types*** eszközbe, és a jobb felső sarokban található ***Select type to add...*** menüpontból válasszuk a gyárilag meglévő típusadatot. Itt két adatot kell majd megadnunk: az új típus nevét vagy azonosítóját, illetve azt a már létező típust, amelyre alapozni akarjuk. Az ***ATFDocument*** típust fogjuk létrehozni, amit értelemszerűen a ***Document*** nevű CMF alapértelmezett típusra fogunk alapozni.

Miután az új típust létrehoztuk, az összes típuslistában látható és elérhető lesz, ideértve a típuseszközöket és a tartalomnézetet, amelyben a típusból új példányokat készíthetünk. Tulajdonképpen bárki, aki rendszergazdai jogosultságokkal rendelkezik az oldalon, mostantól megtalálhatja az új ***ATFDocument*** típust abban a menüben, ahonnan az újonnan létrehozandó új típusokat is kiválaszthatja.

De mi értelme van ennek, ha egyszer az ***ATFDocument*** és a ***Document*** teljesen egyforma? Nos, valójában nem teljesen egyformák, inkább csak azonos tagfüggvényeken és általános osztálymeghatározásokon osztoznak. A típus egyéb jellemzői, mint például a tulajdonságok, az engedélyek és a bőrök (skins) alapértelmezés szerint azonosak ugyan a ***Document*** beállításaival, de bármikor teljesen megváltoztathatják a kinézetet. Így például, ha azt szeretnénk, hogy a ***Document*** fehér alapon feketével jelenjen meg, vitafórum nélkül, az ***ATFDocument*** viszont gesztenye alapon sárga színnel és vitafórummal, ezt ezzel a módszerrel minden további nélkül, néhány kattintással kialakíthatjuk. Azután, amikor a későbbiek során CMF-rendszerünket fejlesztjük, az ***ATFDocument*** típus a ***Document*** típussal együtt önműködően frissülni fog.

A színfalak mögött

Könnyen előfordulhat azonban, hogy olyan típust szeretnénk, amely a meglévő típusoktól teljesen eltérő mezőkkel vagy viselkedéssel bír. Erre is létezik néhány lehetőség, ezek közül a legrugalmasabb (s egyben legnagyobb kihívást jelentő, és legsilányabbul dokumentált) módszer a CMF-szabályoknak megfelelő új Zope-termék létrehozása. Minden Python-csomagnak a csomag gyökérkönyvtárában tartalmaznia kell egy ***__init__.py*** állományt. Ez az állomány akár üres is lehet, de itt helyezhetjük el azokat az utasításokat, amelyek a modul első memóriába töltése során elindulnak. A termékek esetében maga az osztály az `initialize()` tagfüggvény segítségével az ***__init__.py*** belsejében jegyzi be magát a Zope rendszerébe, ami egyetlen, általában `context`-nek nevezett értéket fogad. A lecsupaszított Zope-termék tehát egyetlen ***__init__.py*** állományból áll, ami a következő misztikus `MyProduct`-termékhez hasonlít:

```
import MyProduct
def initialize(context):
  context.registerClass(
    MyProduct.MyProduct,
    constructors=
    ➥(MyProduct.manage_addMyProductForm,
      MyProduct.manage_addMyProduct)
    )
```

A Zope induláskor végignézi a termékeket és a megfelelő összefüggésekkel (`contex`) meghívja az `initialize()` tagfüggvényeket. Az összefüggések a Zope szerzeményező rendszerének a részei, ahol az objektumok tulajdonságait a rendszerben elfoglalt helyük legalább annyira meghatározza, mint az eredeti osztálymeghatározás. A fenti példában a `MyProduct` két létrehozóval (constructor) jegyzi be magát: ez a `manage_addMyProductForm` és a `manage_addMyProduct`. A CMF-típusnak nemcsak a Zope, hanem a CMF rendszerébe is be kell jegyeznie magát, már amennyiben meg akar jelenni a különféle CMF-eszközök között. Termékünk `initialize()` tagfüggvényének ezért a CMF-hez tartozó bejegyzési műveleteket is tartalmaznia kell, azaz a ***__init__.py*** programnak modulokat kell importálnia a CMF-ből. Továbbá minden CMF-típusnak be kell jegyeznie magát a `Products.CMFCore.utils` egyik CMF vonatkozású alaphelyzetbe állító eljárásával. Például a CMF-be épített ***CMFDefault __init__.py*** programja első lépésként megadja azokat az osztályokat, amelyeket az elkövetkezendőkben be fog jegyezni:

```
contentClasses = ( Document.Document
                 , File.File
                 , Image.Image
                 , Link.Link
                 , Favorite.Favorite
                 , NewsItem.NewsItem
                 , SkinnedFolder.
                   ➥SkinnedFolder
)
```

Majd valamennyi osztályhoz megad egy létrehozót:

```
contentConstructors = ( Document.addDocument
     , File.addFile
     , Image.addImage
     , Link.addLink
     , Favorite.addFavorite
     , NewsItem.addNewsItem
     , SkinnedFolder.addSkinnedFolder
)
```

Természetesen minden osztály saját egyedi eszközzel rendelkezhet:

```
tools = ( DiscussionTool.DiscussionTool
        , MembershipTool.MembershipTool
        , RegistrationTool.RegistrationTool
        , PropertiesTool.PropertiesTool
        , URLTool.URLTool
        , MetadataTool.MetadataTool
        , SyndicationTool.SyndicationTool
)
```

Végül a csomag `initialize()` tagfüggvénye, amit egy kicsit lerövidítve adunk közre, eszközök esetében a `utils.ToolInit()`, tartalom esetében pedig a `ContentInit` eljárással bejegyzi az osztályokat. Ezután a visszakapott értékkel meghívja az `initialize(context)` függvényt, ezáltal jegyezve be az új terméket a Zope rendszerébe:

```
def initialize( context ):
   utils.ToolInit(´CMFDefault Tool´,
                  ➥tools=tools,
```

```
                    product_name='CMFDefault',
                    icon='tool.gif',
                    ).initialize( context )

utils.ContentInit( 'CMFDefault Content'
          , content_types=contentClasses
          , permission=AddPortalContent
          , extra_constructors=
            ➥contentConstructors
          , fti=Portal.
            ➥factory_type_information
          ).initialize( context )
context.registerClass(Portal.CMFSite,
      constructors=(
      ➥Portal.manage_addCMFSiteForm,
      ➥Portal.manage_addCMFSite,
                    ))
```

Láthatjuk, hogy az `initialize()` fenti változatának utolsó utasítása igen hasonló a `MyProduct()` példa `initialize()` függvényének utolsó részéhez, ami ékesen bizonyítja, hogy a CMF-típusok valójában néhány további kiegészítéssel rendelkező Zope-termékek.

Érdemes használni a CMF rendszert?

E cikk zárja a Zope mint tartalomkezelői felület körüli vizsgálódásainkat. Körutazásunk a Plone-nal kezdődött, és a CMF-rendszerrel, illetve a CMF-típusokkal zárult. Most, hogy már egy kicsit részletesebben is megismerhettük a CMF rendszerét, eldönthetjük, érdemes-e CMS alapú projektekbe belekezdenünk.

Jó hír, hogy CMF igen hatékony és rugalmas rendszer. Egy ügyes és hozzáértő fejlesztő kezében lehetővé teszi, hogy saját CMS rendszert hozzunk létre a piacon jelenleg elérhető üzleti rendszereknél jóval olcsóbban, ugyanakkor nagyobb rugalmassággal. Minthogy a rendszer teljes egészében a gyors fejlesztésekhez szánt Zope-ra épül, az új típusok létrehozása, a sablonok módosítása és az új szolgáltatások fejlesztése igen egyszerű és gyors.

Ugyanakkor a CMF, akárcsak a legtöbb nyílt forrású program, iszonyú hiányt szenved naprakész, használható leírások terén. Bizonyos vagyok benne, hogy a Plone CMS rendszerének sikere nagyrészben a Plone-hoz tartozó kiváló leírásnak köszönhető.

Tehát amennyiben használni szeretnénk a CMF-et, készüljünk fel egy komoly adag Python-kód átrágására, nem kevés kísérletezgetésre és más CMF-fejlesztők segítségének igénybevételére. Figyelembe véve a CMF központi szerepét a Zope világában, véleményem szerint a CMF-leírások minősége és mennyisége folyamatosan javulni fog. Amíg azonban ez az idő el nem jön, a CMF-fel való munka sok-sok türelmet, forráskódböngészést, próbálkozást és hibajavítást igényel.

A CMF rendszert jelenlegi állapotában – a legnagyobb és legösszetettebb tartalomkezelő rendszerek kivételével – egy kicsit haboznék bármi másra felhasználni.

Ahogy mondják, a CMF rugalmassága és hatékonysága pontosan ebben a méretarányban mutatkozhat meg a leginkább. Röviden: amennyire a CMF nem megfelelő a kisebb munkákhoz, valószínűleg épp annyira jól működik a nagyoknál. Az idő múlásával várhatóan egyre jelentősebb szerepet tölt majd be a nyílt forrású tartalomkezelésben, keretrendszert biztosítva egy saját CMS programrendszer gyors kifejlesztéséhez.

Összefoglalás

A Zope CMF saját CMS-készítéshez használható lenyűgöző keretrendszer. Kétségem sincs afelől, hogy a CMF alatt könnyedén lehet CMS rendszert készíteni, ráadásul lényegesen kevesebb költséggel és erőfeszítéssel, mint amennyit egy teljes értékű üzleti megoldás igényelne. Azt mondják, a CMF ideje még nem érkezett el azok számára, akik nem állnak bensőséges viszonyban vele vagy nem szeretnének jelentős időt pazarolni a megtanulására. Elgondolkodtató, hogy a CMF a Plone által került reflektorfénybe, az a tény pedig, hogy a Zope 3 közel vagy teljesen a CMF-fel egybeépítve jelenik majd meg, azt sugallja, hogy a Zope Corporation mostantól komoly késztetést érez a CMF lenyűgözőbbé és jobban dokumentálttá tételére, mint korábban.

Amennyiben komolyabb programozói tapasztalatokkal rendelkezünk a Python és a Zope területén, szinte biztosan fel tudjuk használni a CMF-rendszert saját típusaink Zope-termék alakú létrehozásához, velük pedig lenyűgöző és érdekes oldalakat készíthetünk magunknak és ügyfeleink számára. Ugyanakkor, amíg a típuskészítő rendszer nem lesz egy kicsit jobban érthető, a Zope-közösségen kívül a CMF nem fogja az őt megillető figyelmet megkapni. A Zope-termékek létrehozása ma már nem tűnik olyan fekete mágiának, mint korábban, és várhatóan a jövőben a CMF-típusok készítésére is ugyanilyen szemmel tekintünk majd.

A következő hónapban hirtelen fordulatot veszünk, és egy másik nyílt forrású CMS rendszert vizsgálunk meg: a Bricolage-t. A Bricolage, ami Mason-, `mod_perl`- és PostgreSQL-alapokra épül, igen szép területet hasított ki magának az elmúlt években, és egyre jelentősebb résztvevője a nyílt forrású CMS-közösségnek.

Linux Journal 2003. augusztus, 112. 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 Izraelben, Modi'in-ben él.

KAPCSOLÓDÓ CÍMEK

A Zope CMF honlapja a ➲ http://cmf.zope.org

Már a honlapon sem tudtam egyszerűen eligazodni, ráadásul semmilyen jó minőségű, hasznos útmutatót sem találtam CMF-témában.

A legjobb, CMF-típusokról szóló bemutatkozó jellegű írást nem a CMF-oldalon, hanem a Plone oldalán, a ➲ http://www.plone.org címen találtam. Például a ➲ http://plone.org/documentation/CMFTypesBook/backtalk_book_view címen fellelhetjük a CMF Types Book írást, amely jól olvasható és számos példát tartalmaz. A The Plone Book 8. fejezete szintén tartalmaz néhány hasznos útmutatót a CMF-típusokról. Ezt a ➲ http://plone.org/documentation/book/8 címen találjuk. Mint mindig, a ➲ http://www.zopelabs.com címen található a ZopeLabs szép számú példakódja és minibemutatója.