

## Zope-termékek készítése

Reuven ebben a hónapban elmeséli, hogyan készíthetünk egyszerű Zope-termékeket, illetve miképpen illeszthetjük be őket honlapunkba.

**M**últ hónapban folytattuk a Zope webfejlesztő környezetben tett kirándulásunkat, és megvizsgáltunk, illetve telepítettünk néhány Zope-terméket. Mint láhattuk, minden termék tulajdonképpen egy Python-objektummodul, amelyet azután egy vagy több példányban létrehozhatunk a Zope-kiszolgálón. Termékek százaikat tölthetjük le a Zope-hoz, kezdve az apró, pehelysúlyú szerencsemondótól egészen a hatalmas és lenyűgöző tartalomkezelő keretrendszerekig (content management framework – CMF).

Számos Zope-pal dolgozó rendszergazda meg is elégszik a Webről letöltött anyagok telepítésével és használatával. Kétségtelen, hogy átlagosan egyszerű honlapigényeink kielégítéséhez bőven található termék a neten; amit pedig nem tudunk termékek segítségével megoldani, az többnyire elég egyszerű ahhoz, hogy megírassuk DTML-ben, a Zope Dynamic Template Markup Language nyelven (erről a témáról bővebben a Linuxvilág 2002 februári számában olvashatunk).

Bármennyire is egyszerű és magától értetődő bizonyos dolgokat DTML-ben megoldani, soha nem olyan teljes és rugalmas, mintha Python alatt készítenénk. Igaz ugyan, hogy a Python-Scripts (és a PerlScripts) megjelenése a Zope-ban számos közepes méretű feladatban szükségtelenné tette termékek készítését, mégis a legtöbb Zope-programozó előbb-utóbb azon kapja magát, hogy valamilyen terméket ír.

Ebben a hónapban megvizsgáljuk, hogyan írhatunk egyszerű Zope-terméket, amit aztán beépíthetünk a honlapunkba. Mint látni fogjuk, igen könnyű a környezet többi részéhez jól illeszkedő Zope-terméket készíteni.

### Egy igen egyszerű termék

A Zope-termékek lényegében Python-modulok. A termékek – mint a múlt hónapban láthattuk – a fő Zope könyvtár *lib/python/Products* alkönyvtárba kerülnek. A Zope a termékeket csak induláskor és újraindításkor nézi meg, így minden új termék telepítése után a Zope-ot újra kell indítanunk.

A telepített termékeket ezután tetszés szerinti számban létrehozhatjuk, minden példányt elhelyezve valahol a honlapszerkezetben. Minden példány egyedi azonosítóval (ID) rendelkezik, amely egyrészt egyértelműen azonosítja őket a könyvtárban, másrészt lehetővé teszi, hogy az objektum eljárásaira hivatkozzunk.

Lehet, hogy kicsit zavarosan hangzik, de emlékezzünk arra, hogy a */foo/bar* URL általában azt jelenti, hogy a webkiszolgáló a *foo* könyvtárban található *bar* állományt adja vissza. Ezzel szemben Zope alatt a *foo/bar* URL azt jelenti, hogy a rendszernek a *foo* objektum *bar* eljárását kell meghívnia. Más szavakkal a *foo/bar* a *foo.bar* kifejezéssé alakul át. Továbbá amikor azt mondjuk, „a *foo* objektum”, valójában azt kellene mondanunk, hogy „a *foo* azonosítójú objektum”. Az azonosító (ID) beállítása nélkülözhetetlen, ha az objektum

eljárásait sikeresen meg szeretnénk hívni.

Termékünkhöz egy *meta\_type* értéket is meg kell adnunk. A *meta\_type* név lesz az a szöveg, amely a */manage* képernyő jobb felső sarkában fellelhető Zope-terméklista *Add* nevű lenyíló menüjében szerepelni fog. Általában ugyanazt a nevet adhatjuk a *meta\_type*-nak is, amit az osztályhoz is használunk, de valami könnyebben érthető azonosítót is használhatunk. Ne feledjük, hogy az *Add* menü listája ASCII sorrend szerint rendezett, ami azt jelenti, hogy a nagy *Z* előbb következik, mint a kis *a*.

Termékünk létrehozásához a következő lépéseket kell megtennünk:

- Megadunk egy külön modulban elhelyezkedő osztályt és a *lib/python/Products* könyvtár alá telepítjük.
- Megadunk egy *\_\_init\_\_* eljárást, amelyben az *id* példányváltozóhoz valamilyen értéket rendelünk.
- Megadunk egy vagy több eljárást, amelyeknek a visszatérési értéke HTML-t tartalmazó szöveg.
- Megadunk egy *meta\_type* osztályváltozót, amely azután termékünk összes példányánál beállítja a *meta\_type* értékét.

Láthatjuk, hogy mennyire egyszerűen hozhatunk létre termékeket – az *1. lista* jól szemlélteti a folyamatot. Itt megadjuk a *helloworld.py*-t, ezt az egyszerű Zope-terméket, amelyet ezután már telepíthetünk, és hajszal híján példányokat is készíthetünk belőle (hamarosan azt is megtudjuk, hogyan lehetünk úrrá ezen a hiányosságon).

Akad még néhány lényeges elem, amit érdemes megvizsgálni *helloworld* osztályunkban. Elsőként mind az osztály, mind az eljárások leírását is tartalmaznak. Soha nem árt leírást készíteni, és az a tény, hogy a Pythonban egy ilyen beépített szolgáltatást találunk, ritka, de figyelemreméltó emlékeztető: a programozók jól teszik, ha hajlandók leírást fűzni a forráskódjukhoz. A Zope ezt az ajánlást kötelezővé teszi azáltal, hogy a rendszerben használt összes eljárás esetében megköveteli a dokumentációs sorok jelenlétét.

A *helloworld* osztályunk két eljárást is megad: az *\_\_init\_\_* és az *index\_html* eljárást. Az *\_\_init\_\_* eljárást a Zope önműködően hívja meg, amikor az objektumunkból példányokat készít, és általában arra használjuk, hogy alapértéket adjunk példányváltozóinknak, illetve a később szükséges jellemzőket is itt adhatjuk meg. Jelen esetben az *\_\_init\_\_* egyetlen példányváltozónak ad értéket (*self.id*), amely lehetővé teszi, hogy objektumunk nyomon követhesse a saját „személyazonosságát”. Ahogy az várható is, az *\_\_init\_\_* eljárás nem arra való, hogy a külső világból hívjuk meg, ezt az eljárást a Zope-nak magának kell meghívnia.

Az *index\_html* eljárás feladata ezzel szemben az, hogy a



1. lista A helloworld.py egy igen egyszerű Zope-termék

```
class helloworld:
    "Ez egy pØlda Zope-termØk,
    a ·helloworld· osztály"

    meta_type = "helloworld"

    def __init__(self):
        "Ez az eljáræs h v dik meg, amikor æj
        ↪helloworld-pØldÆny j n lØtre"
        self.id = id

    def index_html(self):
        "AlapØrtelmezØs szerint ez az eljáræs
        ↪h v dik meg a k nyvtÆrban"
        return """<html>
        <head>
        <title>Hello, world!</title>
        <body>
        <h1>Hello, world!</h1>
        <p>Ez az egyszerß Python-termØk nk
        ↪kimenete</p>
        </body>"""
```

címén keresztül hívjuk meg. Ha a *helloworld* egy példányát a Zope-kiszolgáló fő könyvtárába (/) helyezzük, az *index\_html* eljárást a */helloworld/index\_html* cím segítségével hívhatjuk meg. Az *index\_html* azonban különleges: a sok Apache kiszolgálón használt *index.html* állományhoz hasonlóan – ha nincsen más eljárás megnevezve – alapértelmezés szerint ez indul el. Végül figyeljük meg, hogy az *index\_html* HTML-t ad vissza hívójának. Nem ad vissza állapotkódot vagy bármi mását a HTML-en kívül.

### Mi hiányzik?

A *helloworld.py* tökéletesen szabályos Zope-termék; fel is telepíthetjük a *lib/python/Products* könyvtárba és a Zope nem fog tiltakozni. Ugyanakkor a Zope sajnos mégsem veszi észre, hogy a *helloworld.py* itt helyezkedik el, nem helyezi az *Add* választéklistába sem, és általában figyelmen kívül hagyja az összes munkát, amit a termék megírásába fektettünk. Nyilvánvalóan egy kicsit fel kell hízlalnunk csontváztermékünket, ha a Zope-pal szeretnénk kapcsolatot tartani. Ezt a továbbfejlesztett változatot *smallhello* terméknek nevezzük el.

Első lépésként át kell alakítanunk az egyetlen modulfájlból álló (*helloworld.py*) termékünk szerkezetét teljes értékű Python-csomaggá. A csomag egy könyvtár (*smallhello*) a Python keresési ösvényén (search path), és a *sys.path* változó adja meg. A modulfájlban egy vagy több Python-forrásfájl található. A mi esetünkben a *smallhello* könyvtár két fájl fog tartalmazni: a *smallhello.py* állományt, amely meglehetősen hasonlít a *helloworld.py*-re (lásd a 2. listát) és az *\_\_init\_\_.py*-t (lásd a 3. listát), amely alaphelyzetbe állítja és segít bejegyezni az objektumunkat.

A *\_\_init\_\_.py* fájl először beolvassa a *smallhello.smallhello*-t, amely meghatározza a modul eljárásait és attribútumait. Az *\_\_init\_\_.py* leglényegesebb része azonban, legalábbis a Zope szemszögéből, a beállító (initialize)

eljárás. Miután a Zope megtalálta és beolvasta a *smallhello* t, meghívja a *smallhello.initialize* – értéként átadva neki a *ProductContext* objektumot (amit „context”-nek nevez). Más szavakkal az objektum alaphelyzetbe állítása azt eredményezi, hogy az objektum bejegyezi magát a kiszolgálón.

Az alaphelyzetbe állító függvény meglehetősen egyszerű, pedig a mi változatunk még alapvető hibakezelést is végez (a try–except páros használatával) a dolgok helyes működésének megteremtéséért. A *smallhello* termék mindössze két értéket ad át a *context.registerClass* osztálynak: a *finalhello.finalhello* objektumot, amelyet fel szeretnénk venni, illetve egy *constructor* csoportot, amelyet akkor kell meghívni, amikor új termékpéldányt hozunk létre. Ne feledjük el kitenni a befejező pontot, ha csupán egyetlen elemet adunk át *constructor*ként; ha nem így teszünk, a Zope nem fogja tudni betölteni a terméket.

A *constructors* érték csak az egyik a számos érték közül, amit átadhatunk a *context.registerClass* osztálynak, s amelyekkel testreszabhatjuk objektumunk Zope-bejegyzését. Például átadhatunk egy *icon* (ikon) értéket, ezáltal tudatva a Zope-pal, hogy melyik képet (azaz a csomag könyvtárában található fájlnevet) szeretnénk csomagunk példányai mellett látni a Zope-ban.

### Az Objektum módosítása

A *helloworld.py smallhello.py*-vé változtatása (2. lista) néhány apró módosítást is igényel. Kezdjük egy új eljárás létrehozásával, amely lehetővé teszi, hogy a Zope-termékünkől új példányokat hozzon létre. Hagyományosan az ilyen kezeléshez kapcsolódó (management-related) eljárások a *manage\_* előtaggal kezdődnek, így eljárásunkat *manage\_smallhello*-nak fogjuk elnevezni. Ez ugyanaz az eljárás, mint amit a *context.registerClass*-nak átadott *constructors* csoportban is megneveztünk.

A *smallhello* osztályunkon végzett legjelentősebb változtatás egyben az egyik legkevésbé nyilvánvaló: egy alosztályt készítettünk a Zope-csomag (OFS.SimpleItem csomag) részeként elérhető Zope termék-alaposztályok *OFS.SimpleItem.SimpleItem* osztályából. A *SimpleItem* alosztályaként örökölhető tulajdonságok nélkül számos dolog – például az objektumok „fogd és vidd” alapú áthelyezése – egyáltalán nem úgy fog működni, ahogy elvárnánk. Létezik néhány alaposztály, amelyektől termékünk egymást örökölhet; a *SimpleItem*, mint a neve is mutatja, társai közt a legegyszerűbb és legkönnyebben érthető.

Miközben megváltoztattam a *smallhello.py*-t, úgy döntöttem, hogy további két tartalomkészítő eljárással is megtoldom.

Az egyikük az *other\_html*, az *index\_html*-hez hasonló tartalmat hoz létre – kivéve természetesen, hogy ha nincsen más eljárás megadva, az *index\_html* fog megjelenni; míg az *other\_html* csak akkor látszik, ha az URL-ben közvetlen módon megnevezzük.

Ezenkívül beillesztettem egy *foo\_file* eljárást is, amely azt mutatja be, hogyan adjunk vissza HTML-tartalmat (vagy DTML-t) a merevlemezről. Egy kicsit bosszantó és kiábrándító lehet, ha minden HTML-tartalmat Python-modulfájlokba kell tennünk; így a DTML-fájlokat egyszerűen csak a csomag könyvtárába kell helyezni, a módosításokat viszont már a programtól teljesen függetlenül végezhetjük. Figyeljük meg, hogy ehhez a *Globals* csomagból a *HTMLFile* eljárást be kellett importálnunk.

A *smallhello.py \_\_init\_\_* függvényét úgy módosítottam,

2. lista A smallhello/smallhello.py

```

import OFS.SimpleItem          # Alap leköröse
from Globals import HTMLFile  # "gy hozhatunk
                                # majd be
                                # HTML-fájlokat

class smallhello(OFS.SimpleItem.SimpleItem):
    """ ez az osztály határozza meg a
    smallhello terméket. Lőtezik egy
    alaphelyzetbe állt eljárásunk
    (__init__), egy másik, amely alapesetben
    egy rövid HTML-zenetet jelenít meg, és egy
    harmadik, amely egy HTML-fájl tartalmát
    jeleníti meg."""

    meta_type = 'smallhello'

    def __init__(self, id, title):
        "Smallhello új példányának
        alaphelyzetbe állítása"
        self.id = id
        self.title = title

    def index_html(self):
        "Léssünk némi alapvető tartalom!"
        return """
        <html>
        <head><title>Hello, world!</title></head>
        <body>
        <h1>Hello, world!</h1>
        </body>
        </html>"""

    def other_html(self):
        "Mög egy kis egyszerű tartalom"
        return """
        <html>
        <head><title>More content!</title></head>
        <body>
        <h1>More content!</h1>
        <p>You can define lots of methods if
        you want...</p>
        </body>
        </html>"""

    def foo_file(self):
        "Bemutatjuk, hogyan tudunk tartalmat
        megjeleníteni fájlban"
        return HTMLFile('foo', globals())

    def manage_smallhello(self, RESPONSE):
        "Smallhello hozzáadása a könyvtárhoz."
        self._setObject('smallhello_id',
            smallhello('smallhello_id',
                'smallhello_title'))
        RESPONSE.redirect('index_html')

```

hogy három értéket fogadjon. Ezek: a `self`, az `id` és a `title` (korábban csak a `self` és `id` értékeket használtuk). Az `__init__` függvény minden új `smallhello.py` példány létrehozásakor meghívódik, amit egyébként a `manage_smallhello` hívással érhetünk el. A `manage_smallhello` belsejében a `self._setObject` hívásunk az objektumazonosítót az általános `smallhello_id`-re állítja, kiegészítve a `smallhello_title` címmel. Mivel példánkban az azonosítót beleégettük a kódba, és mivel az azonosítóknak minden könyvtárban egyedieknek kell lenniük, ez azt jelenti, hogy `smallhello` termékből egy adott könyvtárba csak egyetlen példányt helyezhetünk. Sajnos kevés a hely az értékírás és olvasás ismertetésére, azonban a **Kapcsolódó címek** között megemlített példák gyors átfutása könnyen érthetővé teszi, hogyan kell az ilyesmit elkészíteni. A `self._setObject` meghívása után a `manage_smallhello` a felhasználó böngészőjét átirányítja a fő (`index_html`) eljárásra. Itt valamilyen tájékoztatást is megjeleníthetünk a felhasználó böngészőjét objektumunk egy másik eljárására irányítva, én azonban inkább az egyszerűbb utat választottam, és a felhasználókat a lapunk `/index.html` oldalára küldöm. Miután a `smallhello` terméket telepítettük, újra kell indítanunk a Zope-ot. Most már látnunk kell a `smallhello` elemet az **Add** menü alja környékén; kiválasztva Zope-honlapunk `index.html` oldalán találjuk magunkat. Mivel termékünket nem tettük túl felhasználóbaráttá, az URL-t (`index_html`, `other_html` vagy `foo_file`) kézzel kell a böngészőbe bege-

peelnünk. Természetesen semmi akadálya nincsen, hogy ezek a lapok néhány egymásra való hivatkozást tartalmazzanak, illetve a honlaprendszer más lapjai ne ide mutassanak. Mit szólna? Készítettünk egy Zope-terméket!

### Mi hiányzik még mindig?

Ha jelenlegi formájában akarnánk kiadni `simplehello` projektünket, nemigen akarná senki sem használni. A fent említett nehézségeken kívül (például az egyes példányoknál az egyedi azonosítók hiánya) termékünk nem tartalmaz kezelőtáblákat sem (`management tabs`), amelyek a Zope-ot a rendszergazdák számára oly felhasználóbaráttá teszik. A biztonsági jogosultságokat sem kezeli igazán szabványos és egyszerű módon. Ezeket a képességeket is majdnem olyan könnyű telepíteni, mint amelyeket eddig láttunk. Például minden tábla egy-egy könyvtárat jelent, amelyben két név-érték pár helyezkedik el, a címke (`label`) és a művelet (`action`). A címkéhez rendelt értéket látja a felhasználó a képernyőn, míg a művelethez rendelt érték határozza meg, hogy a Zope melyik eljárást fogja meghívni, ha valaki az adott táblára kattint. A táblák telepítéséhez adjunk meg egy `manage_options` csoportot (`tuple`) objektumunkban, amelynek elemei a táblát írják le. Az egyik leglényegesebb elem, amiről eddig még nem beszéltünk: a felhasználói bemenet. Valójában ezt könnyű megoldani, mivel a Zope a HTML-űrlapokat úgy kezeli, mintha az eljárás normál értékei lennének. Például vegyük a következő HTML-űrlapot:

## 3. lista A smallhello/\_\_init\_\_.py

```

# Az osztályfőjl importálása
import smallhello

# eljárás, amely egy smallhello-példányt hoz
# létre
def initialize(context):
    "Termék nkbil egy példányt hozunk létre"

    # Osztályunk (product) bejegyzése a
    # jelenlegi
    # acquisition contextben, ahol megmutatjuk
    # milyen eljárás (vagy eljárások)
    # hívásának meg, amikor valaki egy
    # példányt készít a termék nkbil.

    # A "Boring" példamodulban fellelhető
    # tréning alkalmazzuk, amely kivételeket
    # használ a termék bejegyzése során
    # felmerülő hibák elfogására.

    try:
        context.registerClass(

            # Milyen objektumot adunk hozzá?
            smallhello.smallhello,

            # Milyen eljárást kell meghívni,
            # amikor egy smallhello-példányt
            # szeretnénk készíteni?
            constructors =
                (smallhello.manage_smallhello,)
        )

    except:
        # Ha valami gond van, jelentsük a
        # stderr csatornában (ahogy ez a Boring
        # bemutat termékben is történik)

        # Importáljuk a teljes körű hibakereső
        # adatokat elérhető tevékeni modulokat
        import sys, traceback, string

        # Megállapítjuk, mi volt a gond.
        type, val, tb = sys.exc_info()

        # Ismertetjük a felhasználóval az okot
        sys.stderr.write(string.join(
            traceback.format_exception(type, val,
                tb), ' '))

    del type, val, tb

```

© Kiskapu Kft. Minden jog fenntartva

```

<form action="manage_edit" method="POST">
  <p>id: <input type="text" name="id"></p>
  <p>Title: <input type="text"
    name="title"></p>
  <p><input type="submit"></p>
</form>

```

Miután a *Submit* gombra kattintottunk, termékünk `manage_edit` eljárásának két név-érték párost adunk át `id` és `title` néven. Az eljárást a következőképpen adhatnánk meg:

```
def manage_edit(self, id, title):
```

Ebben az eljárásban az azonos nevű változók felhasználásával lekérhetjük az `id` és a `title` HTML-úrlapelemek értékét.

### Összegzés

A Zope-termékek a DTML-fájloknál sokkal kifinomultabb és fejlettebb lehetőséget kínálnak a Zope-alkalmazások létrehozására. Nagyobb rugalmasságot tesz elérhetővé, ám egyben nagyobb fegyelmet és a rendszer magasabb szintű ismeretét is igényli. A Zope-termékek írásának ismerete olyasmiről, mintha `mod_perl`-modulokat írnánk az Apache-hoz; ez azt jelenti, hogy az alaprendszer teljes egészében a rendelkezésünkre áll. A programozókat igen gazdag API segíti saját Zope-termékük elkészítésében, a jó leírás hiánya azonban sajnos sokakat elriaszt a próbálkozástól. Saját `simplehello` termékünk bemutatja, hogy igen rövid idő alatt pár sornyi kóddal is jelentős és hasznos alkalmazásokat tudunk létrehozni.

*Linux Journal* április, 96. szám



Reuven M. Lerner

([reuven@lerner.co.il](mailto: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 nyílt forrású Zope-terméket a Zope Corporation (Digital Creations) készíti és tartja karban. Bővebb információ a Zope honlapján található <http://www.zope.com>

A Zope egyik tagja MaxM által készített egyszerű kétrészes „kis termék” készítői leírás elérhető [http://www.zope.com/Members/maxm/HowTo/minimal\\_01/source](http://www.zope.com/Members/maxm/HowTo/minimal_01/source) címen.

A Pythonról többet a <http://www.python.org> oldalon olvashatunk. Két jó Python-könyv a *Learning Python* (Mark Lutz és David Ascher az O'Reilly kiadásában) és a *Python Essential Reference* (David M. Beazley és Guido Van Rossum a New Riders kiadásában).