Zope-termékek készítése

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

Múlt hónapban folytattuk a Zope webfejlesztő környezetben tett kirándulásunkat, és megvizsgáltunk, illetve telepítettünk néhány Zope-terméket. Mint láthattuk, minden termék tulajdonképpen egy-egy Python-objektummodul, amelyet azután egy vagy több példányban létrehozhatunk a Zope-kiszolgálón. Termékek százait 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 megírhassuk DTML-ben, a Zope Dynamic Template Markup Language nyelvén (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*** alkönyvtárába 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 használtunk, 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 hajszál 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 leírást 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· osztÆly"

  meta_type = "helloworld"

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

  def index_html(self):
     "AlapØrtelmezØs szerint ez az eljÆrÆs
      ➥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 mást 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 fájlt 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 `smallhell t`, meghívja a `smallhello.initialize`-t – értékként átadva neki a `ProductContext` objektumot (amit „context”-nek nevez). Más szavakkal az objektum alaphelyzetbe állítása azt eredményezi, hogy az objektum bejegyzi 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 constructorként; 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 ikon (`icon`) 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-termékünkből 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 egysmást 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 helyeznünk, 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 lekØrØse
from Globals import HTMLFile   # ´gy hozhatunk
                               # majd be
                               # HTML-fÆjlokat

class smallhello(OFS.SimpleItem.SimpleItem):
    """ ez az osztÆly hatÆrozza meg a
    smallhello termØket. LØtezik egy
    alaphelyzetbe Æll t  eljÆrÆsunk
    (__init__), egy mÆsik, amely alapesetben
    egy r vid HTML- zenetet jelen t meg, Øs egy
    harmadik, amely egy DTML-fÆjl tartalmÆt
    jelen ti meg."""

    meta_type = ·smallhello·

    def __init__(self, id, title):
        "Smallhello œj pØldÆnyÆnak
        alaphelyzetbe Æll tÆsa"
        self.id = id
        self.title = title

    def index_html(self):
        "LÆssunk nØmi alapvetı tartalmat!"
        return """
        <html>
        <head><title>Hello, world!</title></head>
        <body>
        <h1>Hello, world!</h1>
        <p>Welcome to my ·smallhello· Zope
        ➥product</p>
        </body>
        </html>"""

    def other_html(self):
        "mØg egy kis egyszerß 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
         megjelen teni fÆjlb l"
        return HTMLFile(·foo·, globals())

def manage_smallhello(self, RESPONSE):
    "Smallhello hozzÆadÆsa a k nyvtÆrhoz."
    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 megjeleníthetnénk 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 begépelnünk. Természetesen semmi akadálya nincsen, hogy ezek a lapok néhány egymásra való hivatkozást tartalmazzanak, illetve a honlaprendszer más lapjai ne ide mutassanak.
Mit szólsz? 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 eleddig 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 osztÆlyfÆjl importÆlÆsa
import smallhello

# eljÆrÆs, amely œj smallhello-pØldÆnyt hoz
# lØtre
def initialize(context):
  "TermØk nkbıl œj pØldÆnyt hozunk lØtre"

  # OsztÆlyunk (product) bejegyzØse a
  # jelenlegi
  # acquisition contextben, ahol megmutatjuk
  # milyen eljÆrÆs (vagy eljÆrÆsok)
  # h v djanak meg, amikor valaki œj
  # pØldÆnyt kØsz t a termØk nkbıl.

  # A "Boring" pØldamodulban fellelhetı
  # tr kk t alkalmazzuk, amely kivØteleket
  # hasznÆl a termØk bejegyzØse sorÆn
  # felmer lı hibÆk elfogÆsÆra.

  try:
    context.registerClass(

      # Milyen objektumot adunk hozzÆ?
      smallhello.smallhello,

      # Milyen eljÆrÆst kell megh vni,
      # amikor œj smallhello-pØldÆnyt
      # szeretnØnk kØsz teni?
      constructors =
      ➥(smallhello.manage_smallhello,)
       )

  except:
    # Ha valami gond van, jelents k a
    # stderr csatornÆn (ahogy ez a Boring
    # bemutat termØkben is t rtØnt)


    # ImportÆljuk a teljes k rß hibakeresı
    # adatokat elØrhetıvØ tevı modulokat
    import sys, traceback, string

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

    # Ismertetj k a felhasznÆl val az okot
    sys.stderr.write(string.join(
      traceback.format_exception(type, val,
      ➥tb), ··))

    del type, val, tb
```

```
<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-űrlapelemek 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 olyasmi, 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 bemutatta, hogy igen rövid idő alatt pár sornyi kóddal is jelentős és hasznos alkalmazásokat tudunk létrehozni.

Linux Journal április, 96. szám

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 a ➲ 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).

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/).