Dinamikusan előállított naptárak

Jó lenne, ha weboldalunk figyelmeztetni tudná a felhasználókat a közeljövő eseményeire, esetleg az egész cégnek közös, szinkronban tartott naptárat biztosítana? Python alatt iCalendar fájlokat előállítva mindez valósággá válhat.

A múlt alkalommal a ***Sunbirddel***, a ***Mozilla Foundation*** naptárkövetésre alkalmas önálló alkalmazásával ismerkedtünk meg. Mint láttuk, a ***Sunbird*** képes az ***iCalendar*** formátumú naptárak kezelésére. A naptárak a helyi fájlrendszerben is lehetnek, de ***HTTP***-n keresztül távoli kiszolgálóról is lekérhetők. Láttuk azt is, hogy a ***Sunbirddel*** milyen egyszerű egy távoli kiszolgálón található naptárat használni. Egyszerűen be kell írnunk a megfelelő ***URL***-t egy párbeszédpanelen, majd miután a ***Sunbird*** letöltötte az ***iCalendar*** fájlt, az új események máris megjelennek a képernyőnkön. Az interneten már jelenleg is számos különböző távoli, ***iCalendar*** formátumú naptárat találunk, megkeresésük és előfizetésük viszonylag egyszerűen letudható. Persze mindennek csak akkor van haszna, ha már meglévő, a nyilvánosság számára elérhetővé tett naptárat szeretnénk igénybe venni. De mi legyen, ha szervezetünk az ***iCalendar*** formátumra akarja alapozni a belső adatcserét? Hogyan tudunk ***iCalendar*** fájlokat létrehozni és terjeszteni, lehetővé téve, hogy mások is értesülhessenek az őket érintő eseményekről? Ebben a hónapban az ***iCalendar*** fájlokkal kapcsolatos, a kiszolgáló oldalra vonatkozó tudnivalókat tekintjük át, valamint naptáralkalmazások – mint a ***Sunbird*** – általi letöltésre, szervezeten belüli használatra szánt naptárakat fogunk készíteni.

iCalendar fájlok

Ha két számítógép között naptárakat akarunk cserélni, nyilván szükségünk lesz egy szabványra, mely meghatározza, hogy ezeket a naptárakat hogyan kell formázni. A továbbításra használt protokoll jelenleg nincs meghatározva, ám a szabványok és a napi gyakorlat egyaránt arra utal, hogy a ***HTTP*** csaknem kizárólagos szerepet nyert a hasonló átvitelek kezelésében. A naptárcsere formátuma, mely az ***RFC 2445*** dokumentumban található, tükrözi korának színvonalát. Míg egy új naptárformátum kétségtelenül ***XML***-re épülne, az 1998-ban született ***RFC*** még név-érték párokat alkalmaz, az elemeket egyszerű hierarchiába rendezve. Példaként vegyük elő a múlt hónapban, a ***Sunbird*** megismerésekor is látott ***iCalendar*** fájlt:

```
BEGIN:VCALENDAR
VERSION
 :2.0
PRODID
 :-//Mozilla.org/NONSGML Mozilla Calendar V1.0//EN
BEGIN:VEVENT
UID
 :05e55cc2-1dd2-11b2-8818-f578cbb4b77d
SUMMARY
 :Linuxvilág határidő
STATUS
 :TENTATIVE
CLASS
 :PRIVATE
X-MOZILLA-ALARM-DEFAULT-LENGTH
 :0
DTSTART
 :20050211T140000
DTEND
 :20050211T150000
DTSTAMP
 :20050209T132231Z
END:VEVENT
END:VCALENDAR
```

Mint látható, a fájl a `BEGIN:VCALENDAR` címkével kezdődik és az `END:VCALENDAR` címkével végződik. A fájl tetején néhány a teljes naptárra érvényes adat található, ilyen a `VERSION` és a `PRODID`, alattuk azonban már kezdődik is az első és egyben egyetlen esemény, melyet a `BEGIN:VEVENT` és a `END:VEVENT` címke fog közre. Nyilván nem okoz nehézséget elképzelni, hogy nagyobb számú eseményt hogyan tárolna a fájl.
Az ***iCalendar*** rendszeres időközönként megismétlődő események megadását is lehetővé teszi. Egyetlen `VEVENT` bejegyzés is elég tehát ahhoz, hogy jelezzünk egy minden hét hétfő délutánján megtartott értekezletet vagy figyelmeztessük magunkat arra, hogy kedden és pénteken ki kell tennünk a kukát. Minden eseménynek van kezdő és záró időpontja, ez a `DTSTART` és a `DTEND`, az események hossza gyakorlatilag tetszőleges.
Bár a fenti példából nem tűnik ki, az ***iCalendar*** az ismétlődő eseményeknél kivételek megadását is lehetővé teszi. Ha például a hétfő délutáni értekezletre nyaralás közben nem akarunk elmenni, akkor egy `EXDATE` bejegyzéssel jelezhet-

1. kódrészlet static-calendar.py, egyszerű, Pythonban készült CGI program iCalendar fájl megnyitására és HTTP feletti elküldésére

```
#!/usr/bin/python
# A CGI modul beemelése
import cgi
# Az esetleges hibák naplózása
import cgitb
cgitb.enable(display=0, logdir="/tmp")
# Hol van a naptárfájl?
calendar_directory =
'/usr/local/apache2/calendars/'
calendar_file = calendar_directory + 'test.ics'
# content-type fejrész küldése a felhasználó
böngészőjének
print "Content-type: text/calendar\n\n"
# A fájl tartalmának elküldése a böngészőnek
calendar_filehandle = open(calendar_file, "rb")
print calendar_filehandle.read()
calendar_filehandle.close()
```

jük távol maradásunkat. A naptárat megjelenítő alkalmazás ezt követően figyelmen kívül hagyja az adott dátumra eső ismétlődő eseményt.

iCalendar fájlok közzététele

Ha már van ***iCalendar*** fájl a rendszerünkön, ennek közzététele rendkívül egyszerű. Az ***1. kódrészlet*** egy egyszerű, ***Pythonban*** készült ***CGI*** programot tartalmaz, mely megkeres egy ***iCalendar*** fájlt a megadott könyvtárban, majd visszaadja a fájl tartalmát a kérést intéző naptáralkalmazásnak. Aki még nem írt ***CGI*** programot ***Pythonban***, az a fenti példa alapján talán már rájött, mennyire egyszerű is az egész. Az alapvető ***CGI*** szolgáltatások eléréséhez be kell tölteni a ***CGI*** modult. A következő lépés a ***cgitb***, a ***CGI*** visszakövető modul betöltése, amellyel hiba esetén hibakereső adatokat tudunk írni egy fájlba.
A következő a ***text/calendar*** tartalomtípus (***content-type***) fejrész elküldése. Feltételezhetjük, hogy a webes tartalmak túlnyomó részének tartalomtípusa ***text/html*** (HTML formázású szöveg) vagy ***text/plain*** (egyszerű szöveges fájl), amihez különféle ***image/jpeg***, ***image/png*** és ***image/gif*** elemek társulnak. Az ***iCalendar*** szabvány szerint a naptárfájlokhoz ***text/calendar*** típust kell rendelni, még akkor is, ha a ***Sunbird*** és a hozzá hasonló programok a ***text/plain*** formátumot is elfogadják. A program utolsó lépése a naptárfájl tartalmának elküldése, amelyet a helyi fájlrendszerből olvas ki.
Aki még egyáltalán nem foglalkozott webes programozással, annak alighanem tökéletesen értelmetlennek tűnik a fenti példa. Például teljesen eszement dolognak tűnik az, hogy külön programot írunk egy állandó fájl tartalmának elérésére – csakhogy így megtehetjük, hogy a külvilág felől elfedjük a naptárfájl valódi helyét. Persze vannak erre jobb megoldások is, például az ***Apache Alias*** utasításának használata. A programot továbbfejleszthetnénk úgy, hogy a naptárfájl nevét átadott értékként közöljük vele, ám ilyenkor is szükségünk lenne állandó jelleggel rendelkezésre álló, előre elkészített fájlokra.

iCalendar létrehozása

A valódi megoldás, ami persze érdekesebb is, az, hogy az ***iCalendar*** fájlt menet közben, a felhasználó kérésének hatására hozzuk létre. A ***CGI*** program tehát nem egy meglévő ***iCalendar*** fájl tartalmát adja vissza, hanem maga állítja azt elő, majd átadja a felhasználó naptár-ügyfélprogramjának. Első ránézésre ennek megvalósítása egyszerűnek látszik. Végül is, az ***iCalendar*** fájlformátum egyszerűnek tűnik, valószínűleg nem okoz gondot a program összeállítása. Ha viszont jobban megvizsgáljuk, rájövünk, hogy az ***iCalendar*** fájlt létrehozni sokkal nehezebb, mint beszélni róla, különösen, ha ismétlődő eseményeket is meg akarunk adni. Figyelembe véve az ***iCalendar*** szabvány növekvő népszerűségét és a nyílt forrású tervezetek megszámlálhatatlan sokaságát, meglepve tapasztaltam, hogy az ***iCalendar*** a legnagyobb nyílt programozói közösségek részéről is csak minimális figyelmet érdemelt ki. Meglepetésem annak is volt köszönhető, hogy az ***iCalendar*** évek óta létezik, számtalan vállalat és naptárprogram támogatja, kezdve a ***Novell Evolutiontől***, a ***Lotus Noteson*** keresztül egészen a ***Microsoft Outlookig***. Az ilyen nagymértékű támogatottság általában sokféle nyelvre kiterjedő, gazdag választékot eredményez. Először a ***Perl*** környékén tapogatóztam; a ***CPAN*** archívum számos moduljának, köztük a különféle internetes szabványokkal kapcsolatosaknak köszönhetően méltán szerzett elismerést. Furcsa, hogy bár ***iCalendar*** fájlok feldolgozásához több ***Perl*** modul közül is válogathatunk, létrehozásukhoz egyetlen naprakész modul sem létezik. A ***Net::ICal::Libical*** például egy burkoló lett volna a ***C***-ben készült ***libical*** könyvtárhoz, ám utolsó kiadása egy alfa változat előtti csomag. A ***Net::ICal*** a ***ReefKnot*** tervezet része volt, ami a jelek szerint szintén félbemaradt.
Szerencsére egy dán fejlesztő, ***Max M*** (lásd az internetes forrásokat) nemrég úgy döntött, hogy pótolja a hiányosságokat, és készített egy ***iCalendar*** fájlok egyszerű létrehozására használható ***Python*** csomagot. Én minden gond nélkül le tudtam tölteni és telepíteni tudtam a csomagot a gépemre, és jómagam is úgy találtam, hogy rendkívül könnyű vele naptárat készíteni. Korábbi egyszerű kis ***CGI*** programunkkal együtt alkalmazva kiválóan megfelel naptárak létrehozására és közzétételére.

Dinamikus naptár létrehozása

A ***maxm.dk*** oldalról letöltöttem az ***iCalendar*** csomagot, majd telepítettem. Sok korszerű ***Python*** csomaggal ellentétben telepítése nem önműködő, vagyis kézzel kell bemásolnunk rendszerünk ***site-packages*** könyvtárába, ami az én ***Fedora Core 3*** alapú gépemen a ***/usr/lib/python-2.3/site-packages*** volt. Amint a ***2. kódrészletből*** is látható, az újonnan telepített ***iCalendar*** csomaggal ***Calendar*** (naptár) és Event (esemény) típusú objektumokat hoztam létre. Az első teendőm a megfelelő csomagok beemelése volt az aktuális névtérbe:

```
from iCalendar import Calendar, Event
```

Az ***iCalendar*** csomag ***Calendar*** és ***Event*** modulja rendre a teljes ***iCalendar*** fájlhoz és azon belül egy-egy eseményhez tartozik. A ***Calendar*** objektumból tehát egy példányra van szükségünk, míg minden eseményhez egy-egy ***Event*** objektumnak kell tartoznia.

2. kódrészlet dynamic-calendar.py, iCalendar formátumú naptárat előállító program

```
#!/usr/bin/python
# A CGI modul beemelése
import cgi
from iCalendar import Calendar, Event
from datetime import datetime
from iCalendar import UTC # időzóna
# Az esetleges hibák naplózása
import cgitb
cgitb.enable(display=0, logdir="/tmp")
# content-type fejrész küldése a felhasználó
# böngészőjének
print "Content-type: text/calendar\n\n"
# Naptárobjektum létrehozása
cal = Calendar()
# Milyen termék hozta létre a naptárat?
cal.add('prodid',
        '-//Python iCalendar 0.9.3//mxm.dk//')
# Az RFC 2445-nek a 2.0-s változat felel meg
cal.add('version', '2.0')
# Esemény létrehozása
event = Event()
event.add('summary', 'ATF határidő')
event.add('dtstart',

datetime(2005,3,11,8,0,0,tzinfo=UTC()))
event.add('dtend',

datetime(2005,3,11,10,0,0,tzinfo=UTC()))
event.add('dtstamp',

datetime(2005,3,11,0,10,0,tzinfo=UTC()))
event['uid'] = 'ATF20050311A@lerner.co.il'
# Kapjon kiemelt fontosságot!
event.add('priority', 5)
# Az esemény hozzáadása a naptárhoz
cal.add_component(event)
# A naptár utasítása önmaga leképezésére
iCalendar
# fájlként, majd a fájl átadása a HTTP válaszban.
print cal.as_string()
```

Ezután megalkothatjuk a `Calendar` objektumot:

```
cal = Calendar()
cal.add('prodid',
        '-//Python iCalendar 0.9.3//mxm.dk//')
cal.add('version', '2.0')
```

A második és a harmadik sorban, ahol a `cal.add()` eljárást hívjuk meg, módunk nyílik azonosító adatok hozzáadására az ***iCalendar*** fájlhoz. Az elsővel megadhatjuk az ügyfélprogramnak, hogy milyen alkalmazás hozta létre az ***iCalendar*** fájlt. Ez főleg hibakeresésnél hasznos; ha rendszeresen hibás ***iCalendar*** fájlokat kapunk egy megadott programcsomagtól, akkor kapcsolatba tudunk lépni a készítővel vagy a terjesztővel, és jelenthetjük neki a hibát. A második sorban a változatazonosítót adtuk hozzá, amivel azt jelezzük, hogy az ***iCalendar*** előírások melyik változatát követjük. Az ***RFC*** 2445 értelmében ennek a mezőnek 2.0 értéket kell adnunk – már amennyiben követni kívánjuk a dokumentum előírásait.

Most, hogy megvan a naptárunk, adjunk hozzá egy eseményt, amelyet egészítsünk ki egy összegzés sorral; utóbbi az ***iCalendar*** fájlra előfizető személyek naptáralkalmazásában fog megjelenni:

```
event = Event()
event.add('summary', 'ATF határidő')
```

Mint a példafájlnál is láttuk, minden eseményhez három dátum/idő mező tartozik: a kezdés dátuma és időpontja (***dtstart***), a befejezés dátuma és időpontja (***dtend***) és az az időpont, amikor a bejegyzés bekerült a naptárba (***dtstamp***). Az ***iCalendar*** szabvány egy meglehetősen furcsa formátum szerint tárolja a dátumokat és az időpontokat, ám az Event objektum tudja ezeket kezelni, ha ő maga ***datetime*** objektumot kap a normál datetime Python csomagtól. Tehát:

```
event.add('dtstart',
          datetime(2005,3,11,14,0,0,tzinfo=UTC()))
event.add('dtend',
          datetime(2005,3,11,16,0,0,tzinfo=UTC()))
event.add('dtstamp',
          datetime(2005,3,11,0,10,0,tzinfo=UTC()))
```

Megjegyezném, hogy időzónaként mindhárom esetben ***UTC***-t adtam meg. Amikor az ügyfélalkalmazásban megjelenik az ***iCalendar*** fájl tartalma, akkor természetesen a felhasználó a saját időzónája szerint látja a bejegyzéseket, és nem ***UTC*** szerint.

Az eseményeknek létrehozásuk után egyedi azonosítót kell adni. Az egyediség alatt itt valódi, a világ minden számítógépének minden naptárára kiterjedő egyediséget kell érteni. Ez elég fogós dolognak hangzik, de nem kell megijedni tőle. Sokféle megoldás létezik, az egyik az, hogy a létrehozás időbélyegét, az esemény létrehozására használt számítógép ***IP***-címét és egy nagy véletlenszerű számot kombinálunk. Én egy egyszerű ***UID*** előállítása mellett döntöttem, de ha olyan alkalmazást készítünk, amelyet több számítógépen is használni fognak, akkor érdemes alaposabban átgondolni és egységesíteni az azonosítók létrehozását:

```
event['uid'] = 'ATF20050311A@lerner.co.il'
```

A végső lépés az esemény fontosságának, prioritásának megadása, mely 0 és 9 között lehet. A normál, átlagos fontossági szint az ötös, a sürgősebb elemeknek nagyobb, a kevésbé sürgőseknek kisebb fontosságot kell adni:

```
event.add('priority', 5)
```

Az eseményt létrehozása után csatolni kell a naptár objektumhoz, mely jó ideje csak arra vár, hogy végre kezdjünk vele valamit:

```
cal.add_component(event)
```

Szükség szerint további eseményekkel is bővíthetjük a naptárat, csak az ***UID*** mező egyediségére ügyeljünk. Végül ***Calendar*** objektumunkat az `as_string()` eljárás segítségével beírjuk egy ***iCalendar*** fájlba:

```
print cal.as_string()
```

Mivel a `print` alapesetben a szabványos kimenetre ír, a ***CGI*** programok szabványos kimenete viszont a ***HTTP*** ügyfél felé irányul, végeredményként az ***iCalendar*** fájlt a ***HTTP*** kérést indító ügyfélnek küldjük el. ***MIME*** típusként ***text/calendar***-t választottunk, vagyis a ***HTTP*** ügyfél tudni fogja, hogy a kapott adatokat naptárként kell értelmeznie, illetve képes lesz azok helyes megjelenítésére. Ha megvizsgáljuk a kimenetet, láthatjuk, hogy valóban ***iCalendar*** formátumú:

```
BEGIN:VCALENDAR
PRODID:-//Python iCalendar 0.9.3//mxm.dk//
VERSION:2.0
BEGIN:VEVENT
DTEND:20050311T160000Z
DTSTAMP:20050311T001000Z
DTSTART:20050311T140000Z
PRIORITY:5
SUMMARY:ATF határidő
UID:ATF20050311A@lerner.co.il
END:VEVENT
END:VCALENDAR
```

Be kell vallanom, az előzőhöz hasonlóan ez a példa is egy kicsit mesterkélt volt. Igaz ugyan, hogy a naptárat dinamikusan állítottuk elő, de az esemény bele volt drótozva a programba, így módosítása, hozzáadása vagy törlése csak a programozó számára lehetséges. Fogalmazzunk tehát úgy, hogy újabb lépést tettünk az események és dátumok programból történő előállítása felé. A következő az lesz, hogy a dátumokat fájlban vagy akár relációs adatbázisban helyezzük el, és a programmal az adatokat futás közben alakítjuk át.

Összefoglalás

Ebben a hónapban megvizsgáltuk a dinamikus naptárak a ***Python*** egy egyszerű ***CGI*** programba burkolt ***iCalendar*** moduljával végzett előállítását. Ugyanakkor láttuk azt is, hogy milyen korlátai vannak egy olyan naptárnak, amelynek bejegyzéseit a merevlemezen tároljuk. Jobb megoldás lenne, ha az események adatait egy relációs adatbázisban helyeznénk el, amely önmagában is képes a dátumok kezelésére, valamint megfelelő eljárásokat biztosít a felhasználók és a csoportok hozzáférésének szabályozására. A következő alkalommal tovább bővítjük a naptárprogramot, adatait adatbázisból fogja venni, az ***iCalendar*** fájlokat ***PostgreSQL*** táblák alapján fogja előállítani.

Linux Journal 2005. junius, 134. szám

A cikk forrásai: ➲ www.linuxjournal.com/article/8197

Reuven M. Lerner