© Kiskapu Kft. Minden jog fenntartva

OpenACS-sablonok

Reuven bemutatja, hogyan gyűjtenek és adnak vissza adatot, valamint végeznek önműködő hibaellenőrzést az OpenACS sablonjai.

Ebónapban az OpenACS sablonozórendszerét vesszük közelebbről szemügyre (lásd még a ➲ http://openacs.org/doc/openacs-4/templates.html címet), ami sok tekintetben hasonlít a Zope lapsablonjaihoz (Zope Page Templates – ZPT). Az OpenACS sablonjai kicsit kifinomultabbak, hiszen képesek adatokat gyűjteni és visszaadni, illetve lehetővé teszik olyan önműködő hibakezelési feladatok egyszerű megoldását, amelyet egyébként igen nehéz lenne megvalósítani, vagy egyszerűen kihagynánk.

Az OpenACS-sablonok

Az ArsDigita programozói úgy határoztak, itt az ideje a paradigmaváltásnak. Többé nem hívunk meg lapokat a szokásos ***.html*** és ***.adp*** kiterjesztésekkel; helyette ezentúl mindent kiterjesztések nélkül hívunk meg. Ez azért volt megvalósítható, mert az AOLserver képes kikeresni a megfelelő lapot. A */foo* URL esetében például az AOLserver először a */foo.tcl*, majd a */foo.adp* végül a */foo.html* lapokat próbálja megnyitni.
Az OpenACS sablonozórendszer ezt a képességet használja ki, hogy a munkát megoszthassa a két különböző fájl között.
A HTTP-kérelmek létrehozta kimenet általában két különböző fájlon megy át. Elsőként a ***.tcl*** fájl hajtódik végre, elvégezve az adatbázis-lekérdezéseket és beállítva a változókat. Az utolsó sora általában az `ad_return_template` szokott lenni, egy Tcl-függvény, ami a hozzá tartozó ***.adp***-lapot hívja meg. Az ADP a változókat adatforrásként kapja meg.
Minthogy a ***.tcl***- és az ***.adp***-lapokat várhatóan különböző emberek fejlesztik, természetesnek tűnik, hogy nagyon szétválnak, vagy együttműködési nehézségeik lesznek. Az ArsDigita mérnökei ezt a gondot a „page contract” (lapszabvány) létrehozásával próbálták elkerülni. A lapszabvány a ***.tcl*** fájl által várt HTTP-értékek listája, illetve olyan Tcl-változókészlet (ezeket adatforrásnak – data source – nevezzük), amelyeket az ***.adp***-lap megjelenítésekor használunk fel.

Adatforrások

Az adatforrások egyszerű Tcl-változók, csak másképp nevezzük őket. Az ***.adp***-lapon belül az adatforrásokra `@nØv@` formátumban hivatkozhatunk, azaz a változónév köré @ jeleket helyezünk. Amennyiben az adatforrás nincsen meghatározva, a sablon Tcl-veremlenyomat (stack trace) kiírásával leáll, sérelmezve az ismeretlen változót.
Az adatforrások a fájlon belül bárhol megjelenhetnek. Az értékek még azelőtt eljutnak a HTML-lap megfelelő helyére, mielőtt az a felhasználó böngészőjére kerülne, így az adatforrások nemcsak szöveget, hanem képneveket vagy stíluslapértékeket is megadhatnak. Például az alábbi egyszerű OpenACS-sablon a fejlécsorban a felhasználó vezetéknevét jeleníti meg:

```
<master>
     <h2>@first_name@</h2>

     <p>That's you, isn't it?</p>
</master>
```

A `master` tag jelzi, hogy a kérdéses lap nem teljes értékű HTML-kimenet, hanem a helyi mesterlapba kell beszúrni (azaz a ***master.tcl/master.adp*** lappár által meghatározott lapba). A helyi mesterlap pedig a webhely alapértelmezett mesterlapjába (ez általában a ***default-master.tcl*** és ***default-master.adp*** nevű lappár, de állítható jellemzőről van szó) kerül majd be. Így aztán az eredménylapon a következők jelennek meg:

```
az alapØrtelmezett mester teteje
     a helyi mester teteje
          a sajÆt lapunk
     a helyi mester alja
az alapØrtelmezett mester alja
```

Ezzel a módszerrel könnyen készíthetünk egységes kinézettel rendelkező lapokat azonos fej- és lábléccel, amely például menüsort és kapcsolatadatokat tartalmazhat. A mester és az alapértelmezett mesterlapok jelölése sok tekintetben hasonló a Perl alapú `HTML::Mason` sablonozó rendszer autohandlerjeihez.
Ez mind nagyon szép és jó, de hol van a mi `@first_name@` változónk meghatározva? Nos, ezt a ***.tcl***-társlapban tehetjük meg. Csakhogy nem elég egyszerűen a ***.tcl*** fájlban a `first_name` változót beállítanunk. Meg is kell jelölnünk exportálandó adatforrásként azáltal, hogy közvetlenül megnevezzük.
A ***.tcl***-lapok kimenetüket és bemenetüket az `ad_page_contract` függvény értékeiben adhatják meg. Ez többnyire a lap tetején történik meg. Az `ad_page_contract` hatékony módszer olyan lapok létrehozására, amelyek bizonyos bemenetet várnak, és cserébe várhatóan adott kimenetet készítenek. Az `ad_page_contract` meghívása a hihetetlen egyszerűtől a hihetetlen összetettig változhat a ***.adp***-lapsablon igényeinek megfelelően. Például egy a felhasználó nevét valami próbaértékre beállító, majd azt exportáló ***.tcl***-lap valahogy így nézne ki:

```
ad_page_contract {
    A megjegyzØsek Øs a CVS-adatok ide ker lnek.
} {
} -properties {
    first_name:onevalue
}

set first_name "Pr ba keresztnØv"

ad_return_template
```

Az `ad_page_contract` hívás mutatja meg a sablonozórendszernek, hogy a `first_name` változót exportálni fogjuk. Ezután beállítjuk a `vÆltoz` értékét, majd ezt az értéket az `ad_return_template` hívással átadjuk a sablonnak. (Sajná-

© Kiskapu Kft. Minden jog fenntartva

latos történelmi okokból kifolyólag valamilyen jellemzőbb név helyett az adatforrások a `-properties` változón keresztül kerülnek visszaadásra.)
Több változót is átadhatunk a sablonnak, ha további sorokat írunk az `ad_page_contract`-ba:

```
ad_page_contract {
    A megjegyzØsek Øs a CVS-adatok ide ker lnek.
} {
} -properties {
    first_name:onevalue
    last_name:onevalue
}

set first_name "FirstName"
set last_name "LastName"

ad_return_template
```

Listák és sorkészletek (multirow)

Mint láthattuk, az `ad_page_contract` segítségével egyszerűen adhatunk át szöveges adatot a sablonnak. Sok esetben azonban, főként ha adatbázisból kérünk le adatot, értékek listáját szeretnénk átadni. Az OpenACS-sablonok ezt is gond nélkül képesek kezelni. Példaképpen a következő ***.tcl***-lap a rendszer felhasználóinak listáját kéri le és helyezi a „list” adatforrásba:

```
ad_page_contract {
    A megjegyzØsek Øs a CVS-adatok ide
ker lnek.
 } {
 } -properties {
    users:onelist
}

 set users [db_list get_all_users {
    SELECT PE.first_names || ' ' ||
           PE.last_name as users
      FROM Parties PA, Persons PE
     WHERE PA.party_id = PE.person_id
  ORDER BY PE.last_name, PE.first_Names }]

 ad_return_template
```

Láthatjuk, hogy SQL-lekérdezésünk egyetlen ***users*** nevű oszlopot ad vissza. Az OpenACS adatbázis ***felülete*** ezt a ***users*** Tcl-listává alakítja, amit aztán ***users*** adatforrásként exportálhatunk. Mivel a `:onelist` leíróval exportáltuk, az ***.adp***-lap az összes elemén végiglépkedhet:

```
<master>
      <list name="users">
          <li> @users:item@
      </list>
</master>
```

A `<list>` ismétlőszerkezeten belül a benne foglalt tartalom a lista minden egyes elmére végrehajtódik. Az éppen időszerű elemet `@users:item@` formában érhetjük el; a pillanatnyi ciklussorszámot a `@users:rownum@` tartalmazza, végül a jelenlegi ismétlés `@users` formában érhető el.
Amennyiben több adatbázissoron keresztül szeretnénk ismételgetni, a ***.tcl***-lap sorkészleteket (multirow) is képes exportálni.

A sorkészlet az összes visszaadott sort tartalmazza, az oszlopnevekkel együtt. A dolgok Tcl-oldala a következőképpen néz ki:

```
ad_page_contract {
} {
} -properties {
    users:multirow
}

db_multirow users get_info "
    SELECT PE.first_names || ' ' ||
           PE.last_name as name,
            PA.email FROM Parties PA, Persons PE
     WHERE PA.party_id = PE.person_id
  ORDER BY PE.last_name, PE.first_names"

ad_return_template
```

A `db_multirow` függvény három értéket vár: a benépesítendő tömb nevét (amelyet majd adatforrásként exportálunk), a lekérdezés nevét (amit az adatbázis-független ***.xql*** fájllal kiegészítve használhatunk) és a tartalék lekérdezést, amelyet akkor használunk, ha az ***.xql*** fájl nem található. Az ***.adp*** sablonban a következőket írjuk:

```
<master>
  <ul>
  <multiple name="users">
     <li>
     <a href="mailto:@users.email@"
     @users.name@</a>
  </multiple>
  </ul>
</master>
```

Két trükkös dologra hívnám fel a figyelmet. Először is az ismétlő címke neve multiple, annak ellenére, hogy az adatforrást multirow-ként (sorkészlet) exportáltuk. Ha rossz nevet használunk rossz helyen, az bizony nehezen felderíthető hibákhoz vezethet. Még szebb, hogy a `<multiple>` tagokon belüli elemleíró jel a pont (`@users.email@`), holott a `<list>` tagokban kettőspontot használtunk (`@users:item@`). Jómagam folyamatosan elkövetem ezeket a hibákat, és mindig a korábbi működő lapokat nézegetem, hogy lássam, a helyes írásmódot használom-e a megfelelő lapon.

Bemenetek

Eddig csak azzal foglalkoztunk, hogyan adhatunk át az `ad_page_contract` segítségével adatokat a ***.tcl***-lapból az ***.adp***-lapnak. A ***.tcl***-lapok GET- vagy POST-kérelmeken keresztül azonban bemeneteket is tudnak fogadni. Az `ad_page_contract` lehetővé teszi, hogy megadjuk, milyen bemeneteket várhatunk, szükség szerint alapértelmezett értékeket rendeljünk hozzájuk, sőt ellenőrizzük, hogy a bemenet a megfelelő formátumban van-e:

```
ad_page_contract {
} {
    foo
} -properties {
    foo2:onevalue
}

set foo2 "$foo$foo"
```

```
ad_return_template
```

A fenti példában a *.tcl*-lap `foo` nevű értéke várja (akár GET, akár POST üzeneten keresztül a bemenetet). Az értéket aztán a `foo2` nevű új adatforrás létrehozására használja fel, amelybe a `foo` kétszerezett értéke kerül. A `foo`-hoz alapértelmezett értéket is rendelhetünk, ha a Tcl-lista első elemévé tesszük, majd az alapértelmezett értéket a második elembe helyezzük:

```
ad_page_contract {
} {
    {foo "blah"}
} -properties {
    foo2:onevalue
}

set foo2 "$foo$foo"

ad_return_template
```

Ha ezután a lapot a `foo=abc` értékkel hívjuk meg, az „abcabc" karaktereket kapjuk vissza, ha érték nélkül hívjuk meg, akkor „blahblah" lesz az eredmény.
Minden értékhez egy vagy több jellemzőt is rendelhetünk, így korlátozva a bekért adattípusokat. Például levághatjuk a bevezető és befejező szóköz karaktereket a bemenő értékről, vagy biztosíthatjuk, hogy nullánál nagyobb egész számot kapjunk:

```
ad_page_contract {
} {
    sometext:trim
    anumber:naturalnum
}
```

Egy értékben több jellemzőt is használhatunk, ha vesszővel választjuk el őket egymástól:

```
ad_page_contract {
} {
    sometext:trim,nohtml
    {anumber:naturalnum 50}
}
```

A fenti lapszabvány azt mondja: az `anumber`-nek természetes számnak kell lennie, ha pedig nincsen megadva, az alapértelmezett értéke 50. A `sometext` elejéről és végéről el kell távolítani a szóköz karaktereket, és nem tartalmazhat HTML-tagokat. Létezik egy `html` és egy `allhtml` jellemző is, amelyek biztonságos HTML-címkekezelést, illetve az összes HTML-címkét elérhetővé teszik. A lapszabvány még ennél is többet tud. Létrehozhatunk például egy dátumválasztó elemet (widget) az `ad_dateentrywidget` függvénnyel. Így létrehozható a következő *.tcl*-lap:

```
ad_page_contract {
} {
} -properties {
    datewidget:onevalue
}

set datewidget [ad_dateentrywidget datewidget]

ad_return_template
```

A hozzá tartozó *.adp*-lap, amely a dátumelemet jeleníti majd meg, így néz ki:

```
<master>
    <form method="POST" action="date-2">
    @datewidget@
    <input type="submit" value="Send the date">
    </form>
</master>
```

Más szavakkal a HTML-űrlap a dátumelem tartalmát a `date-2`-be küldi, amely tulajdonképpen az eredményt *.adp*-lappal megjelenítő *.tcl*-kód. A *date-2.tcl* az `ad_page_contract`-ban megadhatja, hogy a bejövő `datewidget` érték egyszerű tömböt tartalmaz. Megadhatjuk azonban a `datewidget`-et date típusúnak is, így önműködően négyelemű tömböt kapunk:

```
ad_page_contract {
} {
  datewidget:array,date
} -properties {
  date_month:onevalue
  date_day:onevalue
  date_year:onevalue
  date_full:onevalue
}
  set date_month $datewidget(month)
  set date_day $datewidget(day)
  set date_year $datewidget(year)
  set date_full $datewidget(date)

  ad_return_template
```

Az időadatokat *.adp*-lapunkon (`date-2`) különböző formátumokban jeleníthetjük meg:

```
<master>
    <p>Month: @date_month@</p>
    <p>Day: @date_day@</p>
    <p>Year: @date_year@</p>
    <p>Full text: @date_full@</p>
</master>
```

Megjegyezzük, hogy a dátumelem (date widget) az SQL-lekérdezésekben is tökéletesen elfogadható. Ez jól jöhet, ha dátumokat kell bevinni, vagy ha összehasonlító lekérdezésekben használjuk őket.

Összegzés

A velem dolgozó grafikusok nagyra értékelik a *.tcl*- és *.adp*-lapok szétválasztását. Míg az OpenACS tanulási görbéje elég meredek lehet, a sablonok működésének megértése egyszerre örömteli és érdekes módja a rendszer megismerésének.

Linux Journal 2003. január, 105. szám

Reuven M. Lerner (reuven@lerner.co.il)
Egy kisebb webes és internetes módszerekkel foglalkozó tanácsadó cég tulajdonosa és vezetője. Az ATF honlapon érhető el (➲ http:/www.lerner.co.il/atf/).

© Kiskapu Kft. Minden jog fenntartva