Hívófél-azonosítás Asterisk és Ajax segítségével

Asteriskkel és Ajaxszal egyszerűen megjeleníthetjük a bejövő és kimenő telefonhívásaink információit.

Asterisket használok a telefonhívásaim kezelésére már körülbelül egy éve. Eközben jöttem rá, hogy nagyon sok érdekes dolgot meg lehet csinálni az *Asterisk*, a *VoIP* és egyéb technológiák bevetésével. Az egyik ilyen trükkös dolog például a bejövő hívások telefonszámának valós idejű kiíratása a webböngészőmben. Ehhez nem kell más, csupán *Asterisk*, *Perl*, *CGI*, *HTML*, *CSS*, *SQL*, *XML* és *aszinkron JavaScript*, vagy röviden *Ajax*. Sok különböző alkotóelemet kell összehozni, de sokszor ez teszi érdekessé a megoldandó feladatot.

A programom működése dióhéjban: ha valaki keres bennünket, akkor az *Asterisk* megvárja a hívófél telefonszámát. Ezután az információ a */tmp*-be kerül pár egyéb információ társaságában. Mindezt az *Asterisk* végzi a *tárcsázási szabályok* (*dial plan*) alapján. A böngészőben megnyitott weboldal másodpercenként lefuttat egy *JavaScript* programot.

Ez *XMLHttpRequest* objektumként lekéri az esetleges új hívások információit. Ezt a szerveren futó *CGI szkript* küldi el *XML* formátumban.

A *JavaScript* feldolgozza a kapott eredményt és megjeleníti. Létrehoztam egy *CSS* állományt is, hogy a feldolgozott információ úgy nézzen ki a megnyitott weboldalon, mint egy *vizuális emlékeztető* (*sticky note*).

Ha vége a hívásnak, az *Asterisk* szerver *SQL* adatbázisba lementi a *hívásrekordot* (*CDR – Call Detail Record*). A *CGI* szkript minden kliens oldali lekérdezéskor ellenőrzi az adott híváshoz tartozó hívásrekord meglétét. Ha létezik, akkor befejeződött a hívás és törli az információs állományt a */tmp*-ből. Ennek megfelelően a vizuális emlékeztető is eltűnik, ha a hívó fél letette a kagylót.

Ráadásul a program egyszerre négy hívást képes kezelni és a kimenő hívásokat is jelzi. Jópofa dolog látni, ki van a vonal túlsó végén – legyen az hívó vagy hívott. Mindezt anélkül, hogy az aktuális hívást megszakítva meg kellene kérdezni. Ha a fiaim idősebbek lesznek, ez még jól jöhet.

Ahhoz, hogy működjön a dolog, az *Asterisk*-et megfelelően be kell állítani, hogy a kapott információkat *SQL* adatbázisba mentse le. Alap beállításokkal az *Asterisk* vesszővel elválasztva adja át az információkat.

A probléma az, hogy az egyszerű szöveges hívásrekord nem tartalmazza a hívás egyedi azonosítóját, amellyel a hívás befejezését lehetne észlelni. Az *SQL*-ben tárolt adatsor azonban tartalmazza. Ez azonban nem lehet probléma. Emlékeim szerint az *Asterisk* hívásinformációk *Postgres* adatbázisba mentése elég egyszerű és jól dokumentált (`cdr_pgsql.conf`). Természetesen használhatunk *MySQL*-t vagy *ODBC*-t is.

Az első és legkönnyebb rész: módosítjuk az *Asterisk* tárcsázási szabályát, hogy létrehozzon egy egyszerű fájlt a bejövő és kimenő hívások esetén. Csupán egy sort kell beszúrni, ha megtaláltuk, hogy hova is kell (az alábbi részletet egy sorba írjuk!):

```
exten => s, n, system(echo
➥"IN#${CALLERID(name)}
➥#${CALLERID(number)}#${UNIQUE
➥ID}" >/tmp/panels/
➥cid/${UNIQUEID})
```

Ez a sor létrehozza egy állományt a */tmp/panels/cid* könyvtárban négy mezővel, a *#*-et használva elválasztónak. Természetesen létre kell hozni a */tmp/panels/cid* könyvtárat megfelelő jogosultságokkal, hogy *Asterisk* tudja írni, a *CGI* szkript pedig olvasni és törölni.

Az első mező a *IN* vagy *OUT*, annak megfelelően, hogy bejövő vagy kimenő hívásról van szó. A következő két mező a hívó nevét és telefonszámát tartalmazza, míg az utolsó a hívás egyedi azonosítóját. Ezt a sort a tárcsázási szabályrendszer megfelelő helyére kell beszúrni. Oda ahol már megkaptuk a hívófél telefonszámát, de még nem csörgetjük a telefont.

Ha szeretnénk a ugyanezt megoldani a kimenő hívásokra is, használjuk az alábbi sort:

```
exten => s, n, system(echo
➥"OUT##${EXTEN}#${UNIQUEID}"
➥> /tmp/panels/cid/$
➥{UNIQUEID})
```

Kimenő hívás esetén nyilván nem kapjuk meg a hívó fél számát, így a második mező üres. Tudjuk viszont a hívott számot, melyet az `${EXTEN}` változó tartalmaz a harmadik mezőben.

Mindkét esetben biztosnak kell lennünk, hogy a mellék és a prioritás mezők frissültek (példánkban s és n).

A lényegre koncentrálva lecsupaszítottam a weboldalt az alapokig, ahogy az az első listán látható. Noha egyszerűnek látszik a *HTML* oldal, mégis minden lényeges benne van. Először is betölti a `cid.js` állományban található

1. Lista Leegyszerűsített weboldal

```
<html>
<head>
  <title>CID Test</title>
  <script language=javascript
➥src=http://hostname/cid.js>
  </script>
  <style type="text/css">
    @import "cid.css";
  </style>
</head>
<body>
  <div id="phone1"></div>
  <div id="phone2"></div>
  <div id="phone3"></div>
  <div id="phone4"></div>
  <script>
    start_cid();
  </script>
  Your Content Would Go Here.
</body>
</html>
```

JavaScript forráskódot. Majd az oldalhoz tartozó stíluslapot a `cid.css` fájlból. A stíluslappal rugalmasan konfigurálhatjuk a vizuális emlékeztetők megjelenését. Utána létrehozunk négy `div` részt, `phone1`...`phone4` nevekkel. Ezek a szekciók a hívásinformációk megjelenítésekor válnak majd láthatóvá. Végül az oldal időnként meghívja a `start_cid()` eljárást. Erre később térünk ki.

Habár a ***CSS*** tudásom nem tökéletes, mellékeltem egy példát, amivel már lehet kísérletezni. (*2. Lista*).

Ez a ***CSS*** fájl tovább ésszerűsíthető a közös tulajdonságok összevonásával, ezt azonban az Olvasóra bízom. A stíluslap négy egyenlő távolságra elhelyezett sárga színű vizuális emlékeztetőt eredményez egy kis árnyékkal kiegészítve (*1. ábra*).

Most pedig lássuk a ***CGI*** szkriptünket (*3. Lista*).

Ez a ***Perl*** szkript a ***/tmp/panels/cid*** könyvtárban keres állományokat, természetesen figyelmen kívül hagyva a `.` és `..` bejegyzéseket. Minden egyéb fájlt megnyit és beolvas. A végeredmény egy ***XML*** formátumú állomány lesz, ahogy az a *4. Listán* is látszik.

2. Lista Példa stíluslap (cid.css)

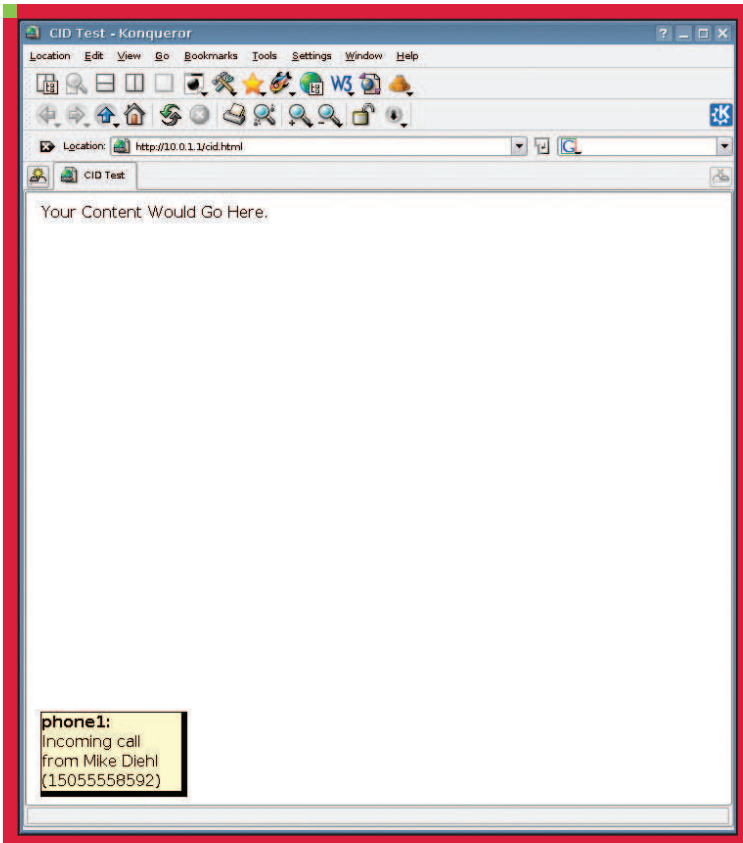
```
div#phone1{
  background: #FFFFCC;
  display: none;
  position: absolute;
  border-top: thin solid
  ➥black;
  border-left: thin solid
  ➥black;
  border-right: 6px solid
  ➥black;
  border-bottom: 6px solid
  ➥black;
  top: 85%;
  left: 2%;
  width: 20%;
  height: 5em;
}
div#phone2{
  background: #FFFFCC;
  display: none;
  position: absolute;
  border-top: thin solid
  ➥black;
  border-left: thin solid
  ➥black;
  border-right: 6px solid
  ➥black;
  border-bottom: 6px solid
  ➥black;
  top: 85%;
  left: 27%;
  width: 20%;
  height: 5em;
}
div#phone3{
  background: #FFFFCC;
  display: none;
  position: absolute;
  border-top: thin solid
  ➥black;
  border-left: thin solid
  ➥black;
  border-right: 6px solid
  ➥black;
  border-bottom: 6px solid
  ➥black;
  top: 85%;
  left: 52%;
  width: 20%;
  height: 5em;
}
div#phone4{
  background: #FFFFCC;
  display: none;
  position: absolute;
  border-top: thin solid
  ➥black;
  border-left: thin solid
  ➥black;
  border-right: 6px solid
  ➥black;
  border-bottom: 6px solid
  ➥black;
  top: 85%;
  left: 77%;
  width: 20%;
  height: 5em;
}
```

3. Lista CGI szkript forrása

```
#!/usr/bin/perl
use DBI;
$dbh = DBI->connect("dbi:Pg:
➥dbname=database",
➥"postgres", "password")
  || die "Can't connect to
  ➥database.\n";
print "Content-type:
➥text/xml\n\n\n";
print "<panels>\n";
check_cid("/tmp/panels/cid");
print "</panels>\n";
exit;
sub  check_cid {
  my($dir) = @_;
  my(@a, $a, $file, $count,
  ➥$top);
  local(*FILE, *DIR);
  opendir DIR, "/tmp/panels/
  ➥cid";
  while ($file = readdir(DIR)) {
    if ($file eq ".") {
    ➥next; }
    if ($file eq "..") {
    ➥next; }
    open FILE, "/tmp/panels/
    ➥cid/$file";
    chomp($line = <FILE>);
    close FILE;
    ($dir, $name, $number,
    ➥$uid) = split("#",
    ➥$line);
```

3. Lista folytatás

```
    $count++;
    if ($dir eq "IN") {
      $html = "Incoming call
      ➥from $name ($number)";
    } else {
      $html = "Outgoing call
      ➥from $name ($number)";
    }
    expire_call($uid);
    print <<EOF
<panel>
  <name>phone$count</name>
  <content>$html</content>
</panel>
EOF
;
  }
}
sub  expire_call {
  my($id) = @_;
  my($sth, $count);
  $sth = $dbh->prepare("select
  ➥count(*) from cdr where
uniqueid=\'$id\'");
  $sth->execute();
  ($count) = $sth->
  ➥fetchrow_array();
  if ($count) {
    unlink("/tmp/panels/
    ➥cid/$id");
  }
}
```



1. ábra A bejövő hívás a weboldalon vizuális emlékeztetőként

4. Lista A CGI szkript által legenerált XML állomány

```
<panels>
<panel>
  <name>phone1</name>
  <content>Incoming call from
  ➥Mike Diehl
(15055558592)</content>
</panel>
</panels>
```

Természetesen az ***XML*** állomány maximum négy `<panel>` blokkot tartalmazhat `phone1`-től `phone4`-ig. A `<content>` blokk a megjeleníteni kívánt szöveget tartalmazza. Minthogy nem egyszerű dolog ***HTML*** kódot XML-be ágyazni, így a `<content>` blokk formázásával nem foglalkoztam túlzottan. Megfigyelhető továbbá a bejövő és kimenő hívások különálló kezelése.

Ha az ***XML*** adott részének legenerálással végeztünk és elküldtük a kliensnek, meghívjuk a `expire_call()` függvényt is. Ezzel leellenőrizzük a hívásrekord meglétét az adatbázisban, hogy véget ért-e a hívás? Az ***Asterisk*** bejegyzi, így ha találatot kapunk, a hívás befejeződött és törölhetjük a fájlt a */tmp/panels/cid* könyvtárból.

A ***JavaScript*** komponens végzi a munka oroszlánrészét és talán ezt a legnehezebb megérteni is. (*5. Lista*). Korábban már volt szó róla, hogy a rendszert a `start_cid()` indításával érjük el. Az eljárás lényege csupán annyi, hogy másodpercenként meghívja a `update_cid()` eljárást. Az `update_cid()` meghívja a `get_from_server()` eljárást, amely böngészőfüggetlen módon kap egy ***XMLHttpRequest*** objektumot. Ezt fogjuk később feldolgozni. Később az `update_cid` meghívja a `clear_panels()` eljárást, mely minden vizuális emlékeztetőt kezdetben üressé tesz és elrejt.

5. Lista JavaScript komponens

```
function start_cid () {
  setInterval("update_cid()",
  ➥1000);
}
function update_cid () {
  var req;
  var xml;
  var panels;
  var count;
  var name;
  var div;
  req = get_from_server();
  clear_panels();
  xml =
req.responseXML.getElementsBy
➥TagName("panels")[0];
  panels = xml.getElementsBy
  ➥TagName("panel");
  for (count=0 ; count <
  ➥panels.length ; count++) {
    panel = panels[count];
    name = panel.getElementsBy
    ➥TagName("name")[0];
    name = name.firstChild.
    ➥nodeValue;
    content = panel.get
    ➥ElementsByTagName
    ➥("content")[0];
    content = content.first
    ➥Child.nodeValue;
    div = document.getElement
    ➥ById(name);
    div.style.display="block";
    div.innerHTML = "<b>" +
    ➥name + ": </b>" +
    ➥content;
    if (div.innerHTML == "") {
      div.style.display=
      ➥"none";
    }
  }
}
function get_from_server () {
  var req;
  if (window.XMLHttpRequest) {
    req = new
    ➥XMLHttpRequest();
  } else if (window.ActiveX
  ➥Object) {
    req = new ActiveXObject
    ➥("Microsoft.XMLHTTP");
  }
  req.open("GET", "/cgi-bin/
  ➥cid.pl", false);
  req.send(null);
  return req;
}
function clear_panels () {
  for (count=1 ; count < 5 ;
  ➥count++) {
    document.getElement
    ➥ById("phone" +
    ➥count).innerHTML = "";
    document.getElement
    ➥ById("phone" +
    ➥count).style.display=
    ➥"none";
  }
  return;
}
```

Láthatóvá akkor válnak, ha lesz valami tartalmuk.
A program további része kicsit bonyolultabb lesz. Felhasználva a korábbi objektumot és a `getElementsByTagName()` eljárást, kapunk egy *XML* objektumot `<panels>` blokkal.
A `getElementsByTagName()` újabb meghívásával különálló `<panel>` blokkokból álló tömböt kapunk. Végül minden, a tömbben szereplő `<panel>` blokkot feldolgozunk. Minden feldolgozott blokk egy folyamatban lévő hívás. Ennek megfelelően az új hívások vizuális emlékeztetőként jelennek meg. Minden `<panel>` blokk egy `<name>` és egy `<contents>` blokkot tartalmaz, melyet megfelelő változókban tárolunk.
A `getElementById()` eljárással lekérdezzünk a panel névvel egyező `<div>` elem azonosítóját (*ID*). Most már minden szükséges információnk megvan, név, tartalom, hely. Nincs más hátra, mint az adott `<div>` láthatóvá tétele és a tartalom hozzárendelése egyszerűen az `innerHTML` attribútum alkalmazásával. A ciklus végén visszatérünk az eljárás elejére és elölről kezdjük.
A bemutatott (választ kér és megjelenít) eljárás másodpercenként és felhasználói beavatkozás, illetve oldal újratöltés nélkül történik.
A felhasználó ebből csak annyit vesz észre, hogy telefoncsörgéskor megjelenik egy vizuális emlékeztető, ha pedig vége a beszélgetésnek, eltűnik.
Amint látható, jól használható nyelv a ***JavaScript***. Sajnos azonban a böngésző oldali támogatása és a fejlesztőeszközök hiánya érezhető. A program fejlesztése közben gyakran omlott össze a böngészőm. Sokszor pedig a tévedésből ***cache***-elt adatokkal és titokzatos hibaüzenetekkel akadtak problémáim. Miután azonban működésre bírtam, meg kellett győződnöm, hogy az általam használt böngészőkben – ***Konqueror*** és ***Firefox*** – hibátlanul működik. Úgy vélem más böngészők alatt is fut, noha nem teszteltem. Minthogy a szoftverfejlesztéseknél jellemzően ***vi***-t használok, járatlan vagyok az integrált fejlesztő környezetekkel (***IDE – Integrated Development Environments***) kapcsolatban. Ennek ellenére ha hallott az Olvasó olyanról, amely ***JavaScript***-et is támogat, szeretettel várom e-mailben.
Most, hogy a program működik, itt az ideje bővíteni. A legnyilvánvalóbb változtatás egy link lesz, mely lehetővé tenné további információk lekérését a hívóról. Ezt persze a címjegyzékemből vagy egy külön adatbázisból. Talán egy kép is megjelenhetne a hívóról, bár elég hosszú időbe telne családtagoktól, barátoktól, ismerősöktől összegyűjteni a képeket. Jó lenne az is, ha egy gombnyomással üzenetrögzítőre irányíthatnám a bejövő hívást. Nem lenne nehéz megvalósítani azt sem, hogy a rendszer jelezze, ha van új üzenetem vagy a barátaim ráérnek csevegni.
Itt van tehát egy jópofa elfoglaltság, amelyben több különféle eszköz és technika van jelen. Vajon mennyit számítana fel a ***Qwest*** egy ilyen webes felület havi díjaként, ha csak a hívófél azonosítás havi 6 dollár?

Linux Journal 2006., 151. szám

Mike Diehl a Sandia National Laboratories-nál dolgozik és munkaidejében hálózatfelügyelő programot fejleszt. Feleségével és két kisfiával Albuquerque-ben, Új-Mexikóban él. Mike a mdiehl@diehlnet.com email címen érhető el.