AUDIENCE AWARENESS

METERING VIEWERSHIP IS AN ESSENTIAL COMPONENT TO THE SUCCESS OF YOUR FLASH VIDEO PROJECT.

BY NELS JOHNSON

Okay, so Flash may now be God to some (in the Eric Clapton sense). One result of this theocracy is that several old Web video questions are new again: Who's watching? How often? How far into the clip? Meaningful answers are of particular interest to Web 2.0 advertisers as ad revenue becomes the last best hope for sites that don't sell retail goods or promote social networks or are already cost centers for non-Web businesses.

In the past, collecting online playback metrics was difficult, expensive and often unreliable. Now, thanks to advances in ActionScript and a software platform known as AJAX, any Web media producer with Flash Studio and basic HTML/Javascript (OK and PHP) skills can build a clip tracking system on a standard Web server at very little expense, The result is accurate, DIY viewership reporting.

The keys to success are: 1) understanding how ActionScript in an embedded SWF file communicates media playback events to JavaScript on the same Web page, and 2) learning how JavaScript can use AJAX to automatically send those time and date-stamped events - perhaps including user geo and browser data - to a simple MySQL database. Armed with such capabilities, you can easily publish private Web pages for advertisers (and other authorized clients) who aggregate, collate and otherwise make sense of all that raw metered data.

The rest of this story provides instructions on how to design and build a basic version of the system described above, including full source code and links to a sample working Web site.

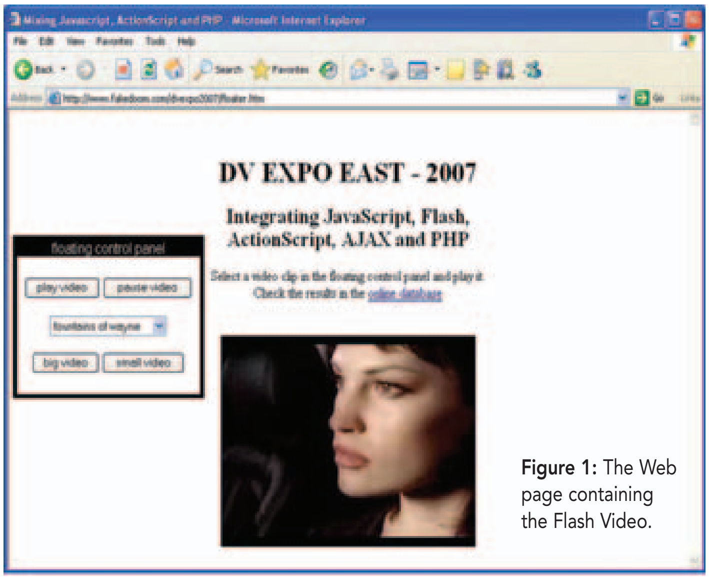
ACTIONSCRIPT/SWF FILE DEVELOPMENT

First the media elements. Open a new .fla project in Flash Studio (I used Version 8 Professional in this example) then instantiate an FLVPlayback component with the following properties: name = vid, video window size = 320x240 (for example), position = upper left, autoplay = false. You can use defaults for the other properties but leave contentPath blank unless you don't care about on-the-fly clip selection (demonstrated below).

Now the ActionScript, which controls the video clip referenced by the FLVPlayback component named vid. You can cut and paste the code below directly into the Actions window for

ACTIONSCRIPT/SWF FILE DEVELOPMENT

```
import flash.external.ExternalInterface;

ExternalInterface.addCallback("playVideo", null,
playVideo);
ExternalInterface.addCallback("pauseResume", null,
pauseResume);

ExternalInterface.addCallback("change_vid", null,
change_vid);
ExternalInterface.addCallback("bigVideo", null,
bigVideo);
ExternalInterface.addCallback("smallVideo", null,
smallVideo);

ExternalInterface.addCallback("playback_complete",
this, end_vid);
ExternalInterface.addCallback("cuepoint_alert", this,
cuepoint_alert);

var myFLV:String;
var listenerObject:Object = new Object();

function playVideo():Void {
  vid.play();
}
function pauseResume():Void {
  vid.pause();
}
function change_vid(new_vid):Void {
  vid.contentPath = new_vid;
  vid.addASCuePoint(10, "elapsed_time");
}
function bigVideo():Void {
  vid.setSize(320,240);
  vid.x = 0;
  vid.y = 0;
}
function smallVideo():Void {
  vid.setSize(160,120);
  vid.y = 60;
  vid.x = 80;
}
listenerObject.complete =
function(eventObject:Object):Void {
  ExternalInterface.call("playback_complete");
};
listenerObject.cuePoint =
function(eventObject:Object):Void  {
  ExternalInterface.call("cuepoint_alert");
};
vid.addEventListener("complete", listenerObject);
vid.addEventListener("cuePoint", listenerObject);
vid.contentPath = "";
```


DV EXPO EAST - 2007

Integrating JavaScript, Flash, ActionScript, AJAX and PHP

Select a video clip in the floating control panel and play it. Check the results in the online database

Figure 1: The Web page containing the Flash Video.

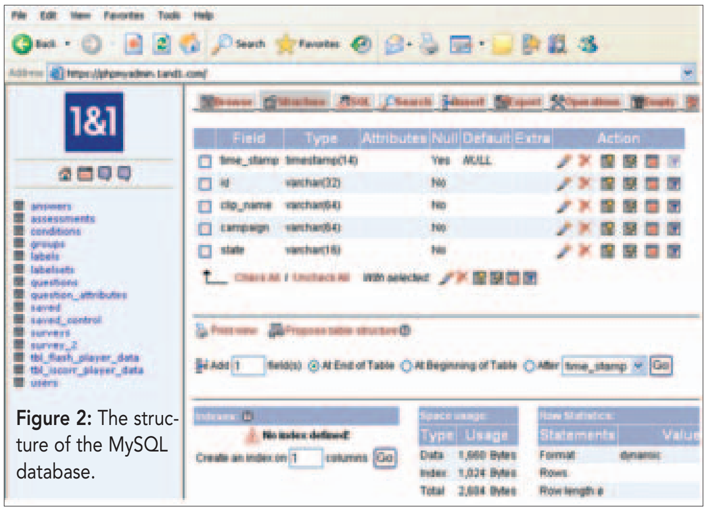

	Field	Type	Attributes	Null	Default	Extra	Action
☐	time_stamp	timestamp(14)		Yes	NULL		
☐	id	varchar(32)		No			
☐	clip_name	varchar(64)		No			
☐	campaign	varchar(64)		No			
☐	state	varchar(16)		No			

Figure 2: The structure of the MySQL database.

```
<?php
session_start();
$chandle =
mysql_pconnect("db696.mydbserver.net",
"dbo635476543", "7Yrf4Wx") or
die("Connection Failure to Database");
$database="db876573524";
mysql_select_db($database, $chandle) or
die ("Database not found.");

if ($_POST["state"] == "partial") {
    $query1="insert into tbl_flash_play-
er_data (";
    $query1= $query1 . "id,";
    $query1= $query1 . "clip_name,";
    $query1= $query1 . "campaign,";
    $query1= $query1 . "state) ";
    $query1= $query1 . "values(";
    $query1= $query1 . "'" .
$_POST["showtime_id"] . "'" . ",";
    $query1= $query1 . "'" .
$_POST["clip_name"] . "'" . ",";
    $query1= $query1 . "'" . $_POST["cam-
paign"] . "'" . ",";
    $query1= $query1 . "'" .
$_POST["state"] . "'" . ")";
} else {
    // assume state is complete here
    $query1 = "update
tbl_flash_player_data set state = '" .
$_POST["state"] . "' ";
    $query1 = $query1 . "where id = '" .
$_POST['showtime_id'] . "'";
}
$result = mysql_db_query($database,
$query1) or die("Failed Query");
mysql_free_result($result);
mysql_close($chandle);
?>
```

the first frame of Layer 1 of your project.

The ActionScript shown above (specific context for each object is available in the Flash Studio help system) breaks down into three functional sections: inclusion and use of the callback facility of *ExternalInterface*, instantiation and exploitation of the *ListenerObject* and integration of these two entities (plus the five other small functions) with the *FLVPlayback* object. Reading through the script several times will give you at least a general idea of these object dependencies and interactions.

The main takeaways are: 1) the *addCallback* methods let JavaScript in the Web page call the internal ActionScript functions, and 2) the *addListener* methods allow ActionScript to tell Javascript (in the Web page) when clip playback is complete or (in this case) 10 seconds have elapsed. It's worth remembering that all this ActionScript comprises functions related to interaction with the clip from *outside* the SWF file. No stock or otherwise traditional SWF controls are enabled or employed. You can publish your SWF (from Flash Studio) without further ado (or accompanying HTML) and keep it handy for use as shown below.

JAVASCRIPT/HTML FILE DEVELOPMENT

The complete text of the HTML file that contains the SWF file that controls the Flash video clip can be procured by viewing (and saving) the source of the Web page browse-able at http://www.fakedoom.com/dvexpo2007/floater.htm. (There is not enough room here for a hard copy of this particular code listing). If you have any trouble saving the HTML file on your desktop, please contact me via e-mail at njohnson@downrecs.com, and I'll send you back a copy. (See Figure 1.)

This HTML file comprises all of the JavaScript necessary to control an FLV video stream selected from a list box, as well as a barebones AJAX routine that writes the playback event data to a MySQL database via a PHP file (source shown below) on the same server. There are no external .JS files. You'll also be able to change the size of the Flash video window (thanks to the callbacks enabled back in the SWF file ActionScript) and run all aspects of the show from a floating control panel (thanks to JavaScript).

WEB SERVER/DATABASE SERVER SETUP

The final piece of software development necessary to record video playback events (resulting from users visiting Web pages and playing Flash video clips) is composing the page that writes to your online database. Story size restrictions prevent a discussion of server hardware configuration, but all of the code presented herein can easily be run on $50-per-month off-the-rack equipment available from the likes of www.godaddy.com and www.1and1.com (host of the demo pages used in this article).

Note that this page employs PHP (the file should have a .php file extension if you use it in production). While not all HTML/Javascript programmers may be comfortable with PHP, the page is short and offers a glimpse of how PHP works with a database - in this case MySQL (see Figure 2). Because PHP runs on the server (JavaScript and ActionScript run on the client), only the output of the page will be shown via your browser's View Source. The PHP code that generates the output, also short and sweet, is shown in the final listing below.

This is all it takes to get started or add value to existing installations. At some point, a client will ask if you can provide such services, and pointing to a working example (however simple) often buys a lot of credibility.

300 WORDS/IMAGES NEEDED

NOT SURE WHAT TO CALL THIS BOX

```
<html>
<head>
</head>
<body>
<center>
<h1>Flash Video Playback Data</h1>
<?php
<?php
session_start();
$chandle = mysql_pconnect("db696.mydbserver.net",
"dbo635476543", "7Yrf4Wx") or die("Connection
Failure to Database");
$database="db876573524";
mysql_select_db($database, $chandle) or die
("Database not found.");
$query1="select * from " . "tbl_flash_player_data";
$result = mysql_db_query($database, $query1) or
die("Failed Query");

//build the html table to present results
$i=0;
echo "<table border=1>";

echo "<tr>";
while ($i < mysql_num_fields($result)) {
  $field_name=mysql_fetch_field($result, $i);
  echo "<th>", $field_name->name, "</th>";
  $i++;
}
echo "</tr>";

while ($thisrow=mysql_fetch_row($result)) { //get one
row at a time
  echo "<tr>";
  $i=0;
  while ($i < mysql_num_fields($result)) {  //print all
items in the row
    echo "<td>", $thisrow[$i], "</td>" ;
    $i++;
  }
  echo "</tr>";
}
echo "</table>";

mysql_free_result($result);
mysql_close($chandle);

?>
</center>
</body>
</html>
```