

Flickr SlideShow lite

(because my grandmother has dialup)

Design Document v1

i. Title

Flickr Slide Show Lite (because my grandmother has dial up)

ii. Author(s)

Paul Russell parussel+slideshow(at)gmail.com (no collaboration planned)

iii. Abstract

Flickr is a great tool for sharing pictures over the Internet, but their flash based slide shows don't work over dial up. This project is to build a lightweight bookmark-able JavaScript/DHTML based slide show that uses Flickr's REST XML API to get picture information and display it over a dial up connection.

iv. What

Slide Show Selection: Users will navigate to a web page (in my Harvard public directory) and have the ability to choose parameters for a Flickr slide show. On this page they will be able to:

1. Provide a username and choose from a list of available picture sets for that user.
2. Submit their selection which will bring them to their slide show of choice.

Slide Show Display: Users can navigate to this page via the slide show selection page, or from a bookmarked URL. The page will use the parameters in the URL (these were constructed by the selection page) to make a request to Flickr for the XML containing the picture list and locations. It will then load the pictures into a slide show and allow users to do the following:

1. Manually navigate forward and backward through the slide show.
2. Have the slide show play with a configurable delay of 1 to 5 seconds between pictures.
3. Optionally display the thumbnails of all the pictures in the slide show.
4. Allow for “Mail this slide show to a friend”.
5. Provide a link to bookmark the slide show.

The most important feature is the ability to store all of the request parameters in the URL and then to be able to translate that URL into a request to Flickr and the results into a light weight slide show. Pictures will be downloaded to the browser one or two at a time to save on bandwidth.

v. How

Basic Goals & Design

As mentioned previously, the goal of this project is to create a Flickr slide show that is light weight from a bandwidth perspective. The elimination of Flash as the slideshow medium is critical to achieving this goal. But we also live in the age of the Slashdot and Digg effect, and so the possibility of heavy load on our hosting server is an issue. This solution should, as much as possible, limit the server side processing requirement for each page displayed.

Flickr Provides a REST API that allows you to query information about images in their servers. The goal is to generate a URL that embeds all the information necessary for a single web page to query. The Slideshow will use this URL both to identify the location of the slideshow viewer as well as to give that viewer all the information required to render a specific set of images in the slideshow. This allows a slideshow to be portable URL format without the need for storage of the slideshow metadata in a server database.

There are challenges with the URL storage format. For one, the URL has a limited length, so we can't embed all of the information about each picture into the URL. Using such an approach would quickly use up the URL character limit on the most robust of web browsers. Another challenge is that while we are attempting to limit bandwidth consumption on the user's computer, we are going to make their web browser responsible for significantly more of the display logic than if we just pushed a preformatted Flash slideshow.

There are 3 main parts to this design

Flickr API: The Flickr API acts as our interface to a database of images and image metadata. Using the RESTAPI we can access information about images. Using the information about those images we can construct URLs to images that we want to display.

AJAX Proxy: Modern Browser Security limits a web page to being able to make an AJAX request only to the domain from which the current page came. This means if I host the slideshow page on www.google.com and I attempt to access the Flickr REST API via AJAX on www.flickr.com. The browser will prevent me from making that request in the name of security. To get around this issue we create a proxy that will be hosted on the local domain. It is designed to accept an HTTP GET request and create a new request with the same parameters to the www.flickr.com domain. The response from the request to Flickr is piped to the output of the CGI. The result is a simple proxy service that is hard coded to work as a local domain AJAX proxy for Flickr's REST API.

Slide Show Webpage: The slide show web page is a simple HTML page that uses cross browser supported CSS and JavaScript to render a slideshow based on the parameters passed in the URL to the slideshow page. Here's how it works:

- 1) It uses JavaScript to parse the parameters on its URL and uses those to determine what image metadata to request from Flickr.
- 2) It then uses AJAX to make a request REST API request for the image metadata for a set of pictures from the FLickr API through our proxy. The result of this request is XML that contains the list of images that belong to the set, with titles, and the details necessary to construct a URL to each image.
- 3) The Response XML is parsed into an array of JavaScript objects that contain all the image metadata.
- 4) A JavaScript Based Slide Show then presents the user with the ability to view one image at a time and navigate through the images in a slide show. It provides both manual forwarded and backward navigation through a list of images as well as a timer based slide show with variable delays between image rendering.

Getting to developing

The remainder of this document describes standards used in this design as well as the specific components that need to be implemented.

Slide Show URL Format

The Slide Show URL needs to contain all the variable information needed to generate a slide show without exceeding the URL size limit of web browser and servers. A safe assumption is that if the URL is < 1024 characters you will be well within the limits of most standards applications of today's age.

The Slideshow URL is made up of the following parts

The base url:

http://<server>/slideshow.html

The parameters required to get images from Flickr:

- method: The Flickr API method to call
- photoseid: The UID of the set of images you want to display
- api_key: The API key for your Flickr API account.

See the Flickr API for more information. Specifically see the section on method=flickr.photosets.getPhotos

The parameters required to set rendering options on the slideshow or reduce the bandwidth consumption of the slideshow.

- userid: The User ID of the person that owns the photos to be displayed
- username: The User name of the person that owns the photos
- photosUrl: The base url to the location of the photos. (flickr lets people configure this)

These parameters can be retrieved from flickr, but their retrieval would require multiple requests to flickr. They are known at the same time the photoseid ID is known so we assume they can be provided as part of the URL

Slideshow URL Parsing

The Slideshow JavaScript should provide a mechanism for parsing the URL parameters and making them readily available as variables or more appropriately constants to the rest of the JavaScript on the page. This is a good example of avoiding the rookie mistake of reinventing the wheel, fire, the light bulb and URL parameter interrogation. Instead we will find someone else that already did the work on this one.

PageQuery(q) creates an object that lets you interrogate the parameters passed along with the page URL. I found this code @ <http://www.eggheadcafe.com/articles/20020107.asp> With minor modifications this code can be made to persist a parameters object instead of creating a new one for each property request.

Using this code we can access the value of any parameter using the function `getProperty(name)` where `name` is the name of the parameter you want.

The AJAX Proxy

The proxy is a simple CGI that takes a GET request, changes the base URL from its location to <http://www.flickr.com/services/rest/> and then sends the response from Flickr back to the original requester. The code to do this is only 6 lines of Perl if you write more, you're over doing it. CGI is a good tool for this, because it's relatively cheap to run, and just about every hosting service supports it. This proxy could be written in Java, C, Ruby, PHP, Python, ASP or any other language capable of making and responding to an HTTP request.

Ajax Request and Response

Once we have the URL parameters parsed and the AJAX Proxy available we can make a call to the Flickr REST API for the images. Fortunately we defined the URL parameters in such a way that the ones needed for Flickr are formatted exactly the same as they are for a Flickr REST request. Fortunately Flickr allows us to pass as much info as we want in a REST request, and it kindly ignores all but the parameters it needs. So We can take the query string (parameter string) on our slideshow URL, and append that to a call to our Proxy. The result will be a properly formatted Flickr API REST request for the image metadata we want.

Again we have a opportunity to avoid reinvention. AJAX is a slick mechanism, but raw AJAX requests require a lot of setup and error handling. Instead we will find a JavaScript package that provides AJAX services to us for Free. Mike West's Data Requestor <http://mikewest.org/archive/datarequestor> provides a few Simple JavaScript functions that encapsulate the code needed to interact with web services over Ajax.

Using Data Requestor you can easily Call the Proxy with the required Flickr parameters, and provide the name of a function that will be called when the AJAX request returns with the resulting XML from Flickr.

For example:

```
var req = new DataRequestor();
req.onload = function (data, obj) { loadImageMetadata(data, 1); }
req.getURL(http://insert\_your\_REST\_url\_here);
```

loadImageMetadata() will be called with the response of the url request

Parsing the Flickr XML Response & The Flickr Picture Object

The Response from the REST API will be formatted in XML. An Example response format can be found at <http://www.flickr.com/services/api/flickr.photosets.getPhotos.html>

Again we will use existing tools to do the XML parsing. XML for <SCRIPT> <http://xmljs.sourceforge.net/website/documentation-classicdom.html> is a powerful JavaScript based XML parser. Using this tool XML from the above example can be easily parsed using the following example code.

```
function loadImageMetadata(data)
{
    var responseDOM = new XMLDoc(data, handleXmlError);

    photoNodes =
    responseDOM.docNode.getElements("photos")[0].getElements("photo");

    // photoNodes is now an array of photo elements

    // loop over the photo nodes and get the details out of each element
    secret = photoNodes[i].getAttribute("title");
}
}
```

Using this parsing code each photo element should be tured into a FlickrPicture Object which is defined as a picture object that stores the metadata for a single picture returned from flickr. Note that the actual image is not stored here, but given the information about the image it is possible to construct a URL to the image.

```
function FlickrPicture(id, owner, secret, server, title, ispublic, isfriend, isfamily)
```

Inside this object you can construct a URL to an image with the following code:
this.mediumImageURL = "<http://static.flickr.com/>" +server+"/"+id+"_"+secret+".jpg";

Each FlickrPicture object should be pushed into stored in a global JavaScript Array variable called "Pictures". This array will be used by the Javascript viewer to determine which images to render.

Page Layout & Image Display

The Slideshow page should contain the following:

- A visible image location that will render the current image in the slide show
IMAGE_1
- A hidden image location that will be used to preload the next logical image to be displayed
IMAGE_2
- A Play button that will start the slide show
- A Stop Button that will pause the slideshow (retaining the current position)
- A << backward button that will let a user manually navigate backward through the slideshow one picture at a time.
- A >> forward button that will let a user manually navigate forward through the slideshow one picture at a time.

Image order is determined by position in the Picture Array. A global variable `currentIndex` is used to keep the `Picture[index]` of the currently displayed image.

The function `displayAllImages()` is a function that uses the `Picture` array and the current image index global variable to determine and set the location `IMAGE_1` and `IMAGE_2` HTML elements to the location. So navigation through the slideshow can be accomplished by creating a function that changes the value of the current image index and then calling `displayAllImages()`;

For example:

```
nextPicture()
{
    if (currentIndex > Pictures.length)
        { currentIndex ++; }
    displayAllImages();
}
```

Obviously, some checking needs to be done to keep from asking for an index that is outside the bounds of the `Picture` Array.

Similarly a `previousPicture()` can be written using a decrement of the index.

The Slideshow effect of having pictures automatically change is accomplished by a use of the `setTimeout` JavaScript function defined here.

http://www.w3schools.com/html/met_win_settimeout.asp

For example you can automatically switch pictures each 3 seconds using the following code.

```
var pictureDelay = 3000; // milliseconds
function play()
{
    nextPicture();
    setTimeout("play()",pictureDelay);
}
```

Creating a stop feature requires that you make play() check to see if the slideshow is still supposed to be running. This can be accomplished with a global Boolean isSlideshowRunning that is checked before taking any play action.

When the last image in the array is displayed, the slideshow should stop.

Creating a Slideshow

With all of the technical details out of the way, we can now goto a slideshow URL, have it parsed for image set details, get the image metadata from Flickr and display the actual images in a slideshow. But that image URL is kinda long and a pain to type. So we're going to add a NEW page that lets us easily create a slideshow URL using the mechanisms we described in the slideshow.

This Create Page will do the following:

- 1) Allow the user to submit a Flickr username
- 2) Lookup that users Flickr UID (which is unfortunately not their username and required to make calls to most of the Flickr API) see <http://www.flickr.com/services/api/flickr.people.findByUsername.html> for more info.
- 3) Use the UID to ask Flickr for that users set of images. See <http://www.flickr.com/services/api/flickr.photosets.getList.html>
- 4) Format the slideshow URL for each set returned from Flickr according to the URL format described at the beginning of this document.

#1 is a simple form submit that triggers an AJAX call to the method described in #2. Of course we will need to use our proxy to make this request.

#2 Uses the Data Requestor and XML for <SCRIPT> mechanisms described above to get an intermediate piece of data that is needed as a parameter for the #3 Flickr API requests.

#3 Uses the Data Requestor and XML for <SCRIPT> mechanisms described above to get a list of sets.

#4 simply formats the resulting set data into urls as required by the slideshow we already developed.

Once you have a URL, its easily rendered as a link on the existing page using code such as:

```
function renderSets()
{
    var links = "";
    for (var i=0; i< PhotoSets.length; i++)
    {
        links += '<a href="'+PhotoSets[i].slideshow+ '&photosUrl=';
        links+= UserPhotosURL +'>'+PhotoSets[i].title+'</a><br>';
    }
    //render the links to a <div id="pagetext"> element on the page
    _document.getElementById('pagetext').innerHTML = links;
}
}
```

The user can then click on any slideshow link and they will be brought to the Slideshow!

File System Layout

This application has several components but very few actual files.

It is broken down as followis

```
slideshow
|
+---- cgi-bin
|
+----- static
|
+----- viewer
```

The cgi-bin contains the proxy.cgi file.

The static directory is responsible for hosting documentation. This document will reside there.

The viewer directory will contain the slideshow create and view html, javascript and css as well as the 3rd party Data Requestor and XML for Script files .js files. While this sounds like a lot of files you might want to break out into separate directories it turns out to be a total of 5 files, so separate directories seemed like overkill.

Details Left up to the user

The mechanics of the slideshow can get very details The following are some areas that the User may want to investigate and can leave out of implement as they see fit. I'm adding this section because I'm considering these options:

- An image delay selector that will let the user select from a 1 to 5 second delay between images.
- A show thumbnails button that would cause thumbnails for all the images to be displayed on the bottom of the page. Clicking on a thumbnail would make the slideshow jump to that image
- Links to the slideshow image owners Flickr homepage, Flickr Page for the image set, and a list of other slideshows available from that user.