Exploring Web Services Exercises

Part I: Some Simple Examples

1. Hacking URL strings
2. Millions of Cats
3. Some random "Random" APIs

Part II: Another Example: Mashup

Think of the Web as a vast system of services that can be accessed, almost like a programming environment, with thousands of function calls that you can use to grab information, which you can then display in whatever format you want. Here's an example of a couple of web services that we can use in conjunction with one another. The links below are to the APIs for two separate web services.

National Weather Service forecasts -
http://graphical.weather.gov/xml/rest.php

Google geocoding -
https://developers.google.com/maps/documentation/geocoding/

Note: The Google APIs now require you to obtain an *API Key* before you can query their APIs. Fortunately this does not require expenditure of funds while doing a reasonable amount of testing. We'll find out how to obtain an API Key in class.

Try this:

Here is a sample URL from the National Weather Service forecasting API:

```
http://graphical.weather.gov/xml/sample_products/browser_interface/ndfdXMLclient.php?
lat=38.99&lon=-77.01&product=time-
series&begin=2004-01-01T00:00:00&end=2013-
04-20T00:00:00&maxt=maxt&mint=mint
```

If you execute this URL in a web browser it will return the forecast maximum and minimum temperatures for a the specified range of dates for the specified location. By changing the dates, times and locations you can get different temperature data.

But there is a problem with changing locations—most humans don't think in latitude and longitude, the format required by this web service. Fortunately there is another web service, Google geocoding service, that converts street addresses and landmarks to geographic latitude/longitude coordinates. The link to that API is above.

On that page there is also a sample URL string for obtaining coordinates for an address:

```
https://maps.googleapis.com/maps/api/geocode/json?
address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&key=XXXXXXXXXXXXXXXXX
```

(You have to replace the X's with your API key for it to work.) It's easy to see where to change address information to get the lat/long for your own address. ***Try it now.***

You can quickly scan the results, find the latitude and longitude, and paste them into the URL for the temperature forecast. This time you'll get the temperature forecast for your own home address.

In both of these examples you notice that the results are returned in a specific format, and contain a lot of detailed information. Scanning for the information, copying and pasting into URLs by hand is tedious and time consuming. But with LiveCode as our programming tool, we can discover more convenient ways to use these web services.

Part III: Accessing Web Services in LiveCode

It is trivial to retrieve results from web services in LiveCode. You simply use the `put` command together with the `URL` keyword. Assume you had pasted one of the web service sample URLs above into a field called "urlfld". To get the result of that request just do this in a button script:

```
on mouseUp
  put fld "urlfld" into tURL
  put URL tURL into fld "data"
end mouseUp
```

That's the easy part, of course. What remains to make a useable utility for accessing a web service is to:

1. Create an interface for easily entering the data the web service needs;
2. Formatting the user inputs into the proper format for embedding into the URL for the web service;
3. Constructing the URL, including the parameters—the name=value pairs we discussed earlier;
4. Parsing the results into an easy-to-read format.

Formatting user inputs. LiveCode provides a simple function to format text strings properly in preparation for inserting them into name=value pairs—the `URLencode()` function.

Forming the URL with its argument list. This requires simply concatenating the base URL for the service together with the name=value pairs, using the LiveCode concatenation operator —**&**. It should end up in the following format:

```
http://web.service.com/?
name1=value1&name2=value2&name3=value3
```

Submit the request. As above, using the `put URL` command.

Parse the returned data to pull out just what you want. The stack we completed in class shows techniques for doing that. See the example stack in the InClass folder on the server, under the Web Services folder. (**Note**: You should be aware that LiveCode provides more powerful and sophisticated ways to read JSON and XML files such as the ones returned by these two web services, but the methods we used are adequate for our purposes here.)

See this link for a discussion of useful LiveCode terms for working with web services.