BYU | BRIGHAM YOUNG UNIVERSITY

Office of Digital Humanities

Back BYU LiveCode Lessons Gateway

Accessing Web Services in LiveCode

Notes:

- This lesson is part of the BYU DigHT 310 Web Services module of instruction. Click here to return to the web services module outline.
- Before reading this material you may find it useful to review a few definitions and LiveCode terms that are useful in this context.

There are many web sites that allow you to look up reference information like weather, zip code for location, dictionaries, and many, many others. The web site www.programmableweb.com is a collection of references to publicly-accessible web service APIs, listing hundreds of services available from any web site. The open secret is that you can use these services not only from any web site but from *any* application that is web-savvy. LiveCode is a web-savvy development environment. Are you thinking what I'm thinking? Yes, you can use these web services to bring these reference resources into your LiveCode stacks!

There are several ways to incorporate results from web services into your stack. They range in complexity from simple to a little tricky, but in general it's not difficult. Let's look at several examples.

Simple CGIs and Other Server-side Scripts

Some URLs when accessed by a web browser, return a specific type of data, and all you have to do in LiveCode is get the URL and "parse" the data returned to find what you want.

Example 1: The U.S. Naval Observatory maintains a web site that lets anybody look up the exact current time:

```
http://tycho.usno.navy.mil/cgi-bin/timer.pl
```

All it does is report the time, and the only variable of course is what the current time is.

Example 2: Every time you access a web server, your web browser reports some basic information about your client computer to the server. A simple server side script can echo back information about the client.

```
http://dev.on-rev.com/myip.irev
```

This server-side script just returns the IP address of the client computer.

Notice that in most cases you're going to get some HTML, XML or other form of marked up text back from the URL. In order to use it in your stack you may have to parse it, or at least set `the htmlText` of a field to it so that it's easy to read.

GET method

Anyone who has used Google or other sites to search the web has seen the GET method in action. The string of seemingly nonsense characters that appear after a URL when you start a search means the search script on the host server is using the GET method to get search data from the client. Many of sites use the GET method for returning information and allow other sites to submit GET requests to have information returned. To use them all you need to know is the base URL and API, or Application Programming Interface, which outlines the parameters required to form a valid request. Here are a few examples:

1. Weather conditions

```
site: http://xml.weather.yahoo.com/forecastrss?
  params
  p=[zip code]&
  w=[WOEID], "Where On Earth ID", a special Yahoo! GeoPlanet code which you can find by entering the zip code
  u=[c or f*]
               * for centigrade or fahrenheit, fahrenheit is default if left blank.

        Parameter list description at: http://developer.yahoo.com/weather/
```

   Scott Rossi, a graphic designer and LiveCode developer has created a very nice example of a stack that uses Yahoo! Weather data in a stack. His stack sends zip code and temperature scale (F or C), and gets back weather--condition image, temp, sky cond., wind dir, wind speed. You can see this stack by opening your LiveCode message box and entering:

```
go stack url "http://dight310.byu.edu/resources/stacks/weather.rev"
```

2. Google search

```
  site: http://www.google.com
  params /search?hl=en&source=hp&q=french+revolution&aq=f&aqi=&oq=
        hl - the search language, defaults to en for USA
        source
        q - the search string, the only required parameter
        aq
        aqi
        oq

        See extensive list at: http://www.seomoz.org/ugc/the-ultimate-guide-to-the-google-search-parameters
```

3. Generate various random text and number sequences - Random.org
4. Online Etymology Dictionary - http://www.etymonline.com
5. Dictionary Aggregator OneLook.com - http://www.onelook.com/

POST method

Web forms that use the POST method work just about the same as forms that use the GET method. The only obvious difference is that the POST method does not show the argument list in the browser's address field like the GET method does. In your LiveCode stack you retrieve data from the web service, not with the `put URL` statement, but by using the LiveCode **`post`** command.

Example sites that use POST:

1. http://www.lipsum.com/ - generates pseudo-Latin text for typesetters and page layout.

2. http://babelfish.yahoo.com - translate from one language to another.

3. http://www.hasbro.com/scrabble/en_US/ - Scrabble dictionary and Word builder

Other Types of RESTful services:

Some web services don't use argument lists at all, but instead require you to build a custom URL string in which the arguments are listed as nodes in the URL. For example, here's an API for accessing BYU Academic Calendar information. (It's not an official API, I just pieced it together using information provided to me by the Office of IT.)

http://livecode.byu.edu/internet/BYU_academic_calendar_API.txt

Try This:

1. Go to random.org and choose one of the other randomizing activities, other than the Sequence Generator from example stack in class. (Or you can choose any web service that you find that uses a GET method in its form.) Figure out how to use one of the other randomizer activities, and construct a GET method request in LiveCode. Make it polished, the way you'd want it if you were having strangers use it. That means the output is well-formatted and readable and the interface is reliable and easy to use.

These pages will help you in completing this assignment:

Steps for incorporating a web service with the GET method.
More on understanding HTML forms.

2. Find a web site that lets you look things up, but that uses the POST method for sending user data to the host server. (You can tell when a form uses POST because no extra characters appear in the URL when you submit your data.) Look at the site's source code ("View Source"—an option in most web browsers.) Find the form that assembles and sends the data to the server and use the information you find to recreate that form in your LiveCode stack.

These pages will help you in completing this assignment:

Steps for incorporating a web service with the POST method.
More on understanding HTML forms.

Back to the Web Services instruction module outline.

Back BYU LiveCode Lessons Gateway