BYU | BRIGHAM YOUNG UNIVERSITY

Office of Digital Humanities

Back BYU LiveCode Lessons Gateway

Web Services Lesson Outline

What is a Web Service?

Simply put, a web service is “a method of communication between two electronic devices over the World Wide Web.” (Source: Wikipedia.)

- They are all over the place. Basically any web site where you submit a request and get data back.
- Examples: google searches, zip code lookup, weather report lookup, online stores, etc.

Published web services APIs

Often providers of web services, such as Google, will publish APIs (Application Programming Interface) documents that describe how to interface with their web services. Good API documentation will describe all of the arguments you can send to a service, with clear examples. It should also tell you what format the data returned by the service is in.

The API Evangelist website is a good place to go to learn about APIs. There is a good introduction and descriptions some of the best APIs available on the internet. If you are interested in producing or consuming web APIs this site is a great place to get started.

There are literally thousands of APIs available on the internet that you can access right now. The problem is, how do you find out what's available and how to access them? Here are a couple of good starting places:

The web site www.programmableweb.com is a repository of information about web services that publish public APIs, listing thousands of known web services. It is a good resource for finding web services that you can incorporate into your projects.

Another good, emerging API search engine is APIs.io.

While the API search sites are good for finding APIs, the sheer number of them is mind boggling. So by way of introduction, here are some interesting examples of web service APIs that are fairly easy to use:

Google Geocoding (a service that changes addresses to latitude/longitude coordinates): https://developers.google.com/maps/documentation/geocoding/

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

Google Street View Images:

https://developers.google.com/maps/documentation/streetview/index

Google Static Maps: https://developers.google.com/maps/documentation/staticmaps/

Random numbers generation: http://www.random.org/clients/http/

Merriam-Webster Dictionary: http://dictionaryapi.com/

Web services "architectures"

There are a number of ways to structure a web service, which impacts how we users access it. One of the simplest and most widely used is called the REST architecture. This is the type of web service we will be exploring in these lessons.

An introductory exercise

This exercise is designed to help you understand how to use web services APIs to construct a valid service request URL. It's the first step toward learning to integrate web services into your LiveCode stacks.

Submitting requests to web services

How do you create an HTML request to submit to a RESTful web service? The key is to create a user interface form that gathers and formats the user input the web service requires.

Web page creators do this with HTML forms. You have doubtless encountered these many times on various web sites.

A simple web page form, from http://www.etymonline.com.

(If you are interested in learning what's behind HTML forms, here is a short lesson for the HTML novice on how to read HTML form tags: http://livecode.byu.edu/internet/aboutForms.php.)

In LiveCode we do the same thing using standard LiveCode control objects. Simply drag fields, buttons, or other objects as needed onto the card to create the form interface.

The whole point of forms is to gather input data from the user, convert it to the proper format, and submit it to the server.

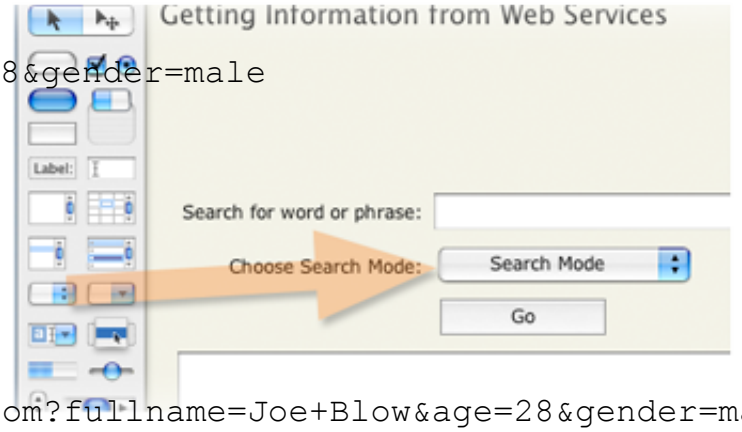
Data collected from forms is typically formatted into a list consisting of name=value pairs, in this format:

```
name1=value1&name2=value2&name3=value3
```

For example:

```
fullname=Joe+Blow&age=28&gender=male
```

Once collected and formatted, this string, called a *query string* or *argument string*, is appended to the end of the web service's URL, separated by a **?** character, like this:



Recreating the form etymologyonline in LiveCode

```
http://some.webaddress.com?fullname=Joe+Blow&age=28&gender=male
```

When submitted to the server in this format, the web service performs whatever calculation or data lookup is called for, then returns data to the client.

Here are some examples web service requests implemented in LiveCode.

Data returned from web services

When a web service responds to a request, the data returned can be formatted in different ways. Sometimes it is returned as HTML, sometimes as plain, unformatted text. But more and more frequently the data returned from web services is formatted in standard formats. The most commonly used formats for sending data from web services are XML and JSON. Here are some introductory lessons to these two formats:

Short introduction to JSON

Short introduction to XML

If you do a lot of work with web service APIs you will need to become familiar with these data exchange formats. For LiveCode developers there are libraries available that make it fairly easy to convert both XML and JSON to and from formats that LiveCode can work with.

No API? Try reverse engineering

Absent a published API it is possible to "mine" HTML source code to discover how to submit data to a web service.

Mining HTML pages for GET method

Mining HTML pages for POST method

Summary

You can access any standard web service from within a LiveCode app. All you need

is:

- The API so you know what kind of data to send.
- Create a form in your stack to gather user input.
- Format the user input into a series of name=value pairs.
- For GET method requests, use the 'put URL urlstring' form.
- For POST method requests, use the LiveCode 'post' command.
- Parse the returned data to display it in a meaningful way.

Assignment

Complete this simple assignment in which you will demonstrate simple web service access in LiveCode.

Back BYU LiveCode Lessons Gateway