Dictionary (https://livecode.com/resources/api/)

Guides (https://livecode.com/resources/guides/)

Lessons (http://lessons.livecode.com/)

Courses (https://livecode.com/products/learn/)

LiveCode Server Guide

Resources (https://livecode.com/resources/)
/ Guides (https://livecode.com/resources/guides/) / LiveCode Server Guide

LiveCode Server Guide

If you want to use LiveCode to create web pages this guide will get you started. The guide explains how to install LiveCode Server on Mac, Windows and Linux servers, create a “Hello World” example, handle user input using Post and Get and debug your scripts.

1. Introduction

The LiveCode Server product brings our English like language to the server environment. The server engine is a separate build of the LiveCode engine with specific syntax and functionality that makes it suitable for use in command-line contexts and, in particular, as a CGI processor.

The principal difference between the server engine and desktop/mobile engines is that it is able to process scripts from text files. There scripts consist of content to output directly interspersed with LiveCode segments. As a result you can now mix LiveCode script with HTML, CSS and Javascript to build webpages in much the same way as you would with other web scripting languages:

Offline (Leave a message)

```
<html>
<head></head>
<body>
<?lc
put "<H1>hello world</H1>"
?>
</body>
</html>
```

Notice that the LiveCode script is contained within special tags. Code within these tags is executed as the page loads allowing you to dynamically generate webpage content. The example above outputs the title “hello world”. The following tags are recognized by LiveCode Server:

```
<?lc ... LiveCode script ... ?>
<?livecode ... LiveCode script ... ?>
<?rev ... LiveCode script ... ?>
```

1.1 File Extensions

As with PHP and other scripting languages HTML/CSS/Javascript/LiveCode scripts are saved as files with a particular file extension. By default, the installation instructions below show you how to configure the server to recognise the .lc file extensions as containing LiveCode script to execute:

```
file.lc
```

You can configure your server to accept the file extension(s) of your choice.

2. Installation

LiveCode Server can be installed on Mac, Windows and Linux servers.

For each supported platform, LiveCode Server is distributed as a single .zip file containing the server engine, drivers and externals as well as release notes. The engine can be run in two separate modes, command line mode or CGI mode. To run the engine in command line mode, unzip the appropriate archive for your platform, then execute the livecode-server binary (e.g. livecode-server.exe on Windows) from the command line, passing the initial script as the first argument. For example:

Offline (Leave a message)

```
[user@~/LiveCodeServer/]$ livecode-server my_script.lc
```

To run the engine in CGI mode, the engine needs to be integrated with the web server software running on your machine. There are various web server packages available, most of which should be compatible with LiveCode Server. In the lessons below we describe how to setup for the most popular package, Apache.

Installation Guide: Main (http://lessons.livecode.com/m/4070/l/36651-how-do-i-install-livecode-server)

The main installation guide links to guides for installing LiveCode server on specific platforms:

Installation Guide: Linux / Apache (http://lessons.livecode.com/m/4070/l/36652-how-do-i-install-livecode-server-on-linux-with-apache)

Installation Guide: Mac OS X / Apache (http://lessons.livecode.com/spaces/lessons/buckets/814/lessons/36653-How-do-I-install-LiveCode-Server-on-OS-X-with-Apache-)

Installation Guide: Windows / Apache (http://lessons.livecode.com/m/4070/l/36654-how-do-i-install-livecode-server-on-windows-with-apache)

Installation Guide: Apache via .htaccess (http://lessons.livecode.com/m/4070/l/36655-how-do-i-install-livecode-server-with-apache-via-htaccess)

3. Getting Started

3.1 Hello World Example

The most basic example using LiveCode Server is to display the text “Hello World” on the screen. To do this create a new file in any text or script editor and add the following line of script:

```
<?lc put "<em>Hello World!<em>" ?>
```

Save the file as:

```
hello.lc
```

Upload the file to your newly configured server and navigate to the file:

Offline (Leave a message)

```
http://www.yourwebsite.com/hello.lc
```

The result:

```
Hello World!
```

To see the hello world example in action along with other practical uses of LiveCode on the server go to our samples page (http://samples.on-rev.com/iframe.irev).

3.2 Using Stacks

The LiveCode Server engine supports loading stacks in a similar manner to that of the desktop engines. Stacks provide the programmer with an additional means to store data and manage code, allowing for improved scope and function name management.

Note: Visual and graphical commands are not supported (e.g. export snapshot).

For a detailed guide on using stacks with LiveCode Server, take a look at our lesson: How do I use stacks with LiveCode Server? (http://lessons.livecode.com/s/lessons/m/4070/l/36656-How-do-I-use-stacks-with-LiveCode-Server-)

3.3 Using Post / Get

There are two main ways of handling user input when developing for the web:

- POST
  Web forms are a common way to get user input. They take the user's entries and pass the data to your scripts using a method called POST. For a detailed step by step guide of how to set up a form and process the POST data with LiveCode script, take a look at our lesson: How do I handle user input using LiveCode Server? (http://lessons.livecode.com/s/lessons/m/4070/l/36650-How-do-I-handle-user-input-using-LiveCode-Server-) You can also try out a practical example here (http://samples.on-rev.com/form.irev).
- GET
  You can also pass data to your scripts via the page URL and the '*query string*'. Take the following URL as a example: http://www.runrev.com/my_script.lc?**key=value&key2=value2**
  The part of the URL in bold, everything after the question mark, is known as the 'query string'. The GET method provides a way to access that information. For a guide and example on how to use this in LiveCode take a look at this lesson: How do I pass information to LiveCode Server scripts? (http://lessons.livecode.com/s/lessons/m/4070/l/36649-How-do-I-pass-

Offline (Leave a message)

information-to-LiveCode-Server-scripts-)
You can also try out a practical example here (http://samples.on-rev.com/get.irev).

3.4 Using Include / Require

LiveCode Server provides '*include*' and '*require*' commands as a way of linking scripts together. This is useful if you want to create libraries which are then used by a variety of other website pages. For example a database library script:

```
http://www.yourwebsite.com/database_library.lc
```

To include the libary:

```
<?lc require "database_library.lc" ?>
```

Learn how we can help you build powerful apps faster and more easily:

I'm Learning to Code

/ For Fun / For Career / For Education (https://livecode.com/try-livecode-i

I'm Switching Tools

/ Looking for a better way to build apps (https://livecode.com/try-livecode

The example below shows the inclusion of the LiveCode database library and the use of two commands contained within it:

```
<?lc
require "database_library.lc"
?>
<html>
<head></head>
<body>
<?lc
databaseConnect()
put databaseShowData()
?>
</body>
</html>
```

include: Includes the given script each time it is called.
require: Only includes the given script if it has not already been included or required.
For more information and a detailed example of these two commands in use see the lesson: How do I add Multiple LiveCode Files in LiveCode Server? (http://lessons.livecode.com/s/lessons/m/4070/l/36662-How-do-I-add-Multiple-LiveCode-Files-in-LiveCode-Server-)

Offline (Leave a message)

3.5 Debugging / Errors

When developing on the server it is vital that you receive as much information on script errors as possible. You to tell LiveCode Server to output error messages in a variety of different ways.

The error mode specifies the action the engine takes when an error occurs. The errorMode property can be set dynamically to one of 4 types:

- **inline** – Indicates that errors should be output to the browser.
- **stderr** – Specifies that the error should be written out to stderr.
- **quiet** – Indicates that nothing is output anywhere when an error occurs.
- **debugger** – For information only and indicates that the script is being run in 'remote debug' mode. This is only relevant to the on-rev engine.

For an example of how to use this see the lesson: How do I display errors when using LiveCode Server? (http://lessons.livecode.com/s/lessons/m/4070/l/6914-How-to-display-errors-when-using-LiveCode-Server)

3.6 Additional Tutorials and Examples

AJAX Tutorial: How Do I Use AJAX with LiveCode Server? (http://lessons.livecode.com/s/lessons/m/4070/l/14117-How-Do-I-Use-AJAX-with-LiveCode-Server-)

AJAX Example: Dynamically filter the list using AJAX (http://samples.on-rev.com/ajax.irev)

Database Example: LiveCode Server database example (http://samples.on-rev.com/database.irev)

Looking for the LiveCode open source project?

Open Source (http://livecode.org)

Looking for the LiveCode FileMaker plugin?

LiveCode for FM (https://filemaker.livecode.com)

(https://www.tagged/livecode)

Developers-

Sitemap (/sitemap)

Offline (Leave a message)