#contentWrapper #fs, #sidebarContent #fs, #contentWrapper div [id * = 'myExtraContent'], #sidebarContent div [id * = 'myExtraContent'] {display: block;}

Kermith's workshop (https://translate.googleusercontent.cc depth=1&hl=en&prev=search&pto=aue&rurl=translate.goog

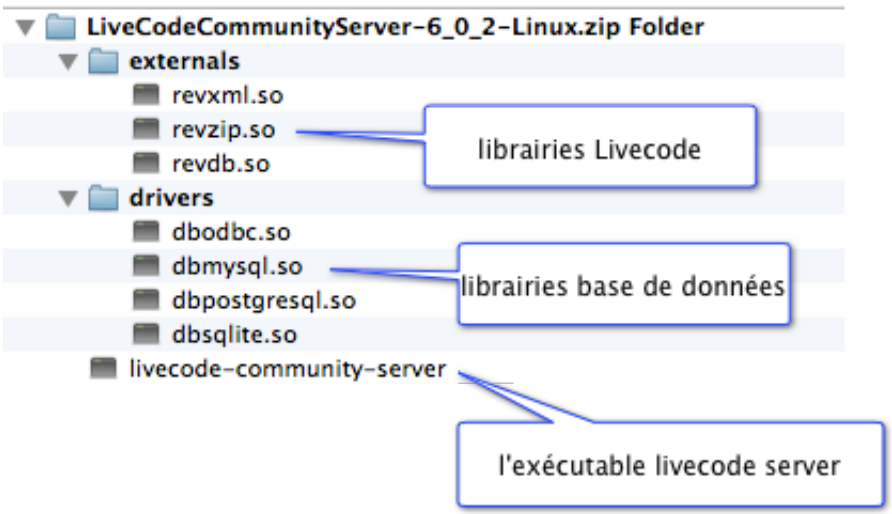
The other way to see supervision ...

LiveCode Server on Linux

The Community version of LiveCode brings a lot of new features over the days. For a few days now, you will find the Community server version on the LiveCode site http://downloads.livecode.com/livecode/server/6_0_2/ (https://translate.googleusercontent.com/translate_c?depth=1&hl=en&prev=search&pto=aue&rurl=translate.google.com&sl=fr&sp=nmt4&u=http://dow
. Using this tool will allow you to use the LiveCode language on a web server in cgi mode. It is even possible to use the code located in the stacks of your programs. On the other hand, the graphic interface of the batteries will not be taken into account.
I propose in this article, the installation of a LiveCode server in a Debian server with Apache.

Installing the LiveCode Server module

Get the LiveCodeCommunityServer-6_0_2-Linux.zip zip file. Unzip this file, you will have the following tree structure:



Livecode Server Tree

I chose to copy this entire tree structure to the / usr / local / livecodeserver folder.

Apache prerequisites

We need to modify the Apache configuration and check if the mod_cgi, mod_actions and mod_alias modules are loaded. Load the Apache modules if you haven't already done so.

```
a2enmod actions
a2enmod cgi
a2enmod alias
```

Next, let's modify the Apache configuration. For my example, I preferred the solution of dedicating a / usr / local / livecode folder for the web server. Let's create the folder.

```
mkdir / usr / local / livecode
```

Let's add the livecode.conf file in the /etc/apache2/conf.d folder

```
Alias / livecode / usr / local / livecode /
<Directory "/ usr / local / livecode">
    MultiViews Index Options
    AllowOverride None
    Order allow, deny
    Allow from all
    AddHandler livecode-script .lc
    Livecode-script / livecode-cgi / livecode-server action
</Directory>

<Directory "/ usr / local / livecodeserver">
     ExecCGI Options
     Order allow, deny
     Allow from all
</Directory>

ScriptAlias / livecode-cgi / livecode-server / usr / local / livec
odeserver / livecode-community-server
```

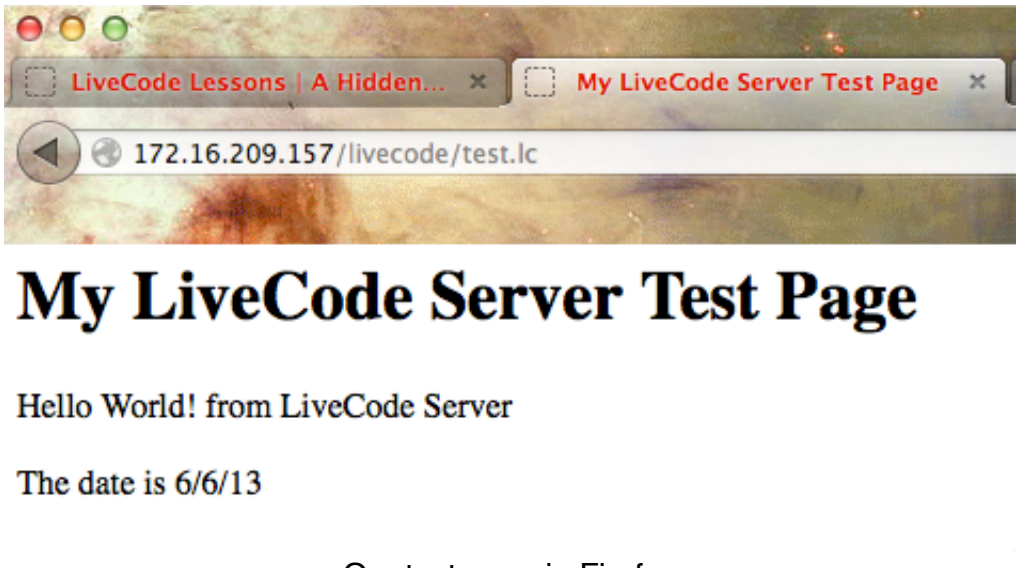
Restart the apache server to take this new configuration into account.

```
apache2 restart service
```

Server test

We will create a test page named test.lc in the / usr / local / livecode folder. the <? lc?> tags are used to insert the LiveCode commands. We will use the put command to send the line in the browser and the date function indicating today's date. These commands are identical in every respect to the commands of the Desktop version, which is the strength of this programming environment.

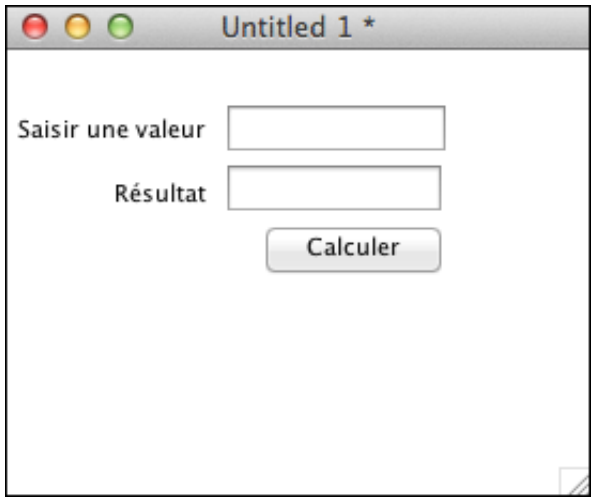
```
<html>

<head>
     <title> My LiveCode Server Test Page
</head>

<body>
     <h1> My LiveCode Server Test Page
<? lc
     put "<p> Hello World! from LiveCode Server </p>"
     put "<p> The date is" && the date & "</p>"
?>
</body>

</html>
```



Our test page in Firefox

Use of a stack (Stack)

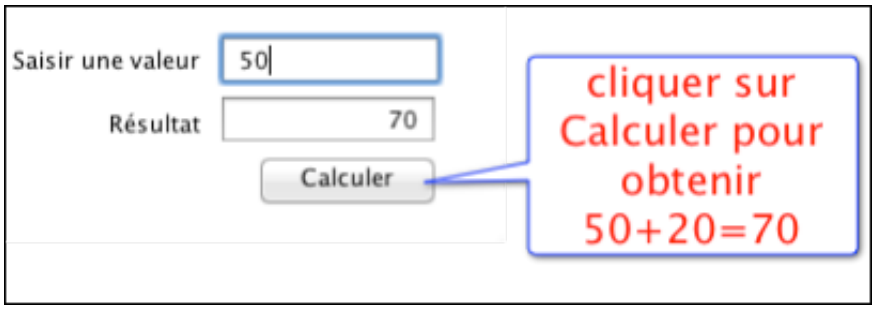
In this chapter, we will discover how to use a stack developed for a thick client type application and transform it into a thin client web application. Let's start by creating our heavy customer. We will use a standard stack and card, an input area, a label for the result and a button to execute the action. This program will have the complex task of adding 20 to the number entered! The important thing is to show how you can reuse code for a web application.



The "thick client" application

The simple application code. The following code will be placed in the card handler.

function calculation
local parameter tvaleur
put parameter into tvaleur
add 20 to tvaleur **return** tvaleur **end** calcul This code will be placed in the button handler.
on mouseUp **put** calcul (the field "TxtValeur") into field "Result" **end** mouseUp Save the stack with the name test_stack.livecode



How the app works

Now we are going to put our stack into operation in our web server. Copy the file in livecode format to the / usr / local / livecode folder. We can reuse the code contained in the card handler. The graphical interface of the card will be replaced by an HTML form. To use the battery code, use the following command:

```
start using stack "test_stack.livecode"
```

Let's create the html file entry_stack.lc

```
<html>

<head>
     <title> Stack usage project </title>
</head>

<body>
<h1> HMI interface </h1>
<? lc
    if $ _POST ["form_submitted"] is true then
        put "<p> The entry is" && $ _POST ["value"] & "</p>"
          if $ _POST ["value"] is not "" then
               start using stack "test_stack.livecode"
               put "<p> The calculation result () is" && calcul ($ _

          else
               put "<p> You forgot to enter a value </p>"
          end if
    else
    ?>
                <form action = "./ grip_stack.lc" method = "POST">
        <p> Enter a value: <input type = "text" name = "Valeur" valu

        <input type = "hidden" name = "form_submitted" value = "true

        <p> <input type = "submit" value = "Calculate" /> </p>
                </form>
        <? lc
    end if
?>
</body>

</html>
```

The page uses the calculation function of the test_stack.livecode stack. The big advantage of this solution is that you can reuse your code as much for Desktop, Mobile or Server applications.

How the web page works

Next article, the use of a database. On your keyboard.