REDCapCon 2020  
  
Beginning REDCap Development

25-Sep-2020

* Philip Chase (Assistant Director, University of Florida - CTSI)
* Kyle Chesney (Application Developer Analyst II, University of Florida - CTSI)
* Luke Stevens (Data Management Coordinator, Murdoch Children's Research Institute)

# Contents

[Contents 1](#_Toc51835900)

[Development Environment 2](#_Toc51835901)

[Environment Setup 2](#_Toc51835902)

[Additional Notes 2](#_Toc51835903)

[External Modules 2](#_Toc51835904)

[File System 2](#_Toc51835905)

[Examples 2](#_Toc51835906)

[Module Creation Process 3](#_Toc51835907)

[Concept 3](#_Toc51835908)

[Design 3](#_Toc51835909)

[Build and Test 3](#_Toc51835910)

[Deploy 3](#_Toc51835911)

[Demonstration of Creating a Module 4](#_Toc51835912)

# Development Environment

## Environment Setup

You need an instance of REDCap you can use for your development and testing. One way to achieve this is to use Docker:

Check out Andy Martin's REDCap Con 2020 video on redcap-docker-compose.

For more guided instruction take a look at this guide from Philip Chase and Kyle Chesney:  
<https://ctsit.github.io/redcap_external_module_development_guide/emd101>

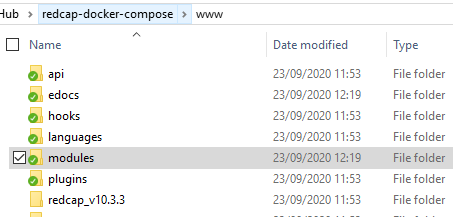
## Additional Notes

* Don't worry if the installation of REDCap takes a long time – 20min or more!
* You do not need to perform any of the SQL steps on the installation page.
* Your REDCap is accessible in your browser at <http://localhost:1935/>
* Your database is accessible using phpMyAdmin at <http://localhost:1935/phpmyadmin/>
* With the release of "granular administrator privileges" in 10.1.0 you may need to run the following SQL statement to grant your admin user permission to set administration privileges:  
  UPDATE redcap\_user\_information SET admin\_rights='1' WHERE ui\_id=2;

# External Modules

## File System

You will now find the REDCap application installed within your redcap-docker-compose directory under www/



There is a top-level modules/ directory into which any modules you download from the REDCap Repo will be installed. You can create your own there too (although it is possible to specify an alternative location).

## Examples

More guidance on developing REDCap external modules, including examples, is available here:

<https://ctsit.github.io/redcap_external_module_development_guide/guide_for_devs>

Check out the REDCap Repo too. You can visit the modules' GitHub sites – they're public – and check out the code there. You can also always just download some to your local instance!

# Module Creation Process

## Concept

* Get an idea! Identify some problem or some improvement that might be made.
* Is there a way to achieve the end without resorting to a module?
* Is there a module that does what you want already?
* Is it worth the effort?
  + Sometimes it can be just for the learning experience. Or for fun!
  + Some institutions do not permit use of external modules.

## Design

* Understand the hook functions - where and when they’re executed.
* What methods to use?
* RTFM ☺ Find the documentation pages within the REDCap Control Centre.

## Build and Test

* Use an IDE with syntax highlighting and ideally with a debugger like xdebug.
* Learn to use your browser's dev tools for inspecting page elements and network traffic, and testing JavaScript (including errors displayed in the console!).

## Deploy

* Use version control software e.g. git/GitHub (which you already used for redcap-docker-compose!) even if you don't plan to release your modules.
* Find out what's involved in getting modules Copy to prod server (staging env first) (institution processes – who?)
* Release to community?

# Demonstration of Creating a Module

<https://github.com/lsgs/redcap-con-2020-beginning-module-dev>

Demo project – and module concept.

Module Creation:

1. Module directory – redcapcon2020\_v0.0.0
2. Required files – starting with minimal content:

**README.md – just some brief descriptive text**

Technically not required, but provide info and guidance to users on what the module does and how to use it.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Demo Module for REDCapCon 2020

Luke Stevens, Murdoch Children's Research Institute https://www.mcri.edu.au

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## Summary

Click the button to look up the max of the value entered and all previously entered values.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**config.json – required elements only**{

"name": "REDCapCon 2020 Demo Module",

"description": "A demo module for REDCapCon 2020.",

"namespace": "MCRI\\REDCapCon2020",

"authors": [

{

"name": "Luke Stevens",

"email": "luke.stevens@mcri.edu.au",

"institution": "Murdoch Children's Research Institute"

}

]

}

**Module class file – note namespace, use and extend (to inherit framework methods etc.)**

Class name and filename must match. Namespace must match config.json<?php

/\*\*

\* REDCap External Module: REDCapCon2020

\* Click the button to look up the max of the value entered and all previously entered values.

\* @author Luke Stevens, Murdoch Children's Research Institute

\*/

namespace MCRI\REDCapCon2020;

use ExternalModules\AbstractExternalModule;

class REDCapCon2020 extends AbstractExternalModule

{

}

Activate the module at system level and for project.

Hook function: redcap\_every\_page\_top():

Illustrating inclusion of custom JavaScript in a page.

Start with just   
 public function redcap\_data\_entry\_form\_top($project\_id, $record=null, $instrument, $event\_id, $group\_id=null, $repeat\_instance=1) {

?>

<script type="text/javascript">

$(document).ready(function(){

alert('It works!');  
 });  
 </script>

<?php

}

Permission required in config.json:  
 ,

"permissions": [

"redcap\_data\_entry\_form\_top"

]

Process of building up bit by bit – now let's add a click handler to our button

$(document).ready(function(){

$('#DemoModuleLookup')

.on('click', function() {

console.log('It works!');

});

});

Next the module settings for the lookup and destination fields – set them for the project:

"project-settings": [

{

"key": "lookup-field",

"name": "Lookup values from this field",

"required": true,

"type": "field-list"

},

{

"key": "set-field",

"name": "Set the max value to this field",

"required": true,

"type": "field-list"

}

]

Additional module page as end point for asynchronous call to server: value\_lookup\_ajax.php.   
Notes:

* $module object provided by framework
* All this really does is call a method in our module class and pass back the result

<?php

error\_reporting(0);

header("Content-Type: application/json");

try {

if (is\_null($module) || !($module instanceof MCRI\REDCapCon2020\REDCapCon2020)) { throw new Exception('No module object'); }

$result = $module->lookupMaxValue($\_POST['value\_entered']);

} catch (Exception $ex) {

http\_response\_code(500);

$result = $ex->getMessage();

}

echo json\_encode(array('result'=>$result));

The function to find the maximum value:

Note:

* Use $this->getProjectSetting('setting') to read module configuration
* Use of \REDCap::getData() developer method for reading record data

public function lookupMaxValue($thisValue) {

$lookupField = $this->getProjectSetting('lookup-field');

// read the existing data for the lookup field

$previousValues = json\_decode(\REDCap::getData(array(

'return\_format' => 'json',

'fields' => $lookupField

)), true);

// loop through all previous values and find the max

$max = intval($thisValue);

foreach ($previousValues as $rec) {

$max = (intval($rec[$lookupField])>$max)

? intval($rec[$lookupField])

: $max;

}

return $max;

}

The completed redcap\_every\_page\_top() function to bring it all together.

Note:

* $this->getUrl() not just a direct reference to the file
* Check the setup before trying to do anything – be defensive!
* See how PHP variables can get written into the JavaScript that's outputted to the page

public function redcap\_data\_entry\_form\_top($project\_id, $record=null, $instrument, $event\_id, $group\_id=null, $repeat\_instance=1) {

// get url for the lookup page

// it's not the directory path e.g. /modules\_dev/redcapcon202\_v0.0.0/value\_lookup\_ajax.php

// but provided by the module framework e.g. /redcap\_v?.?.?/ExternalModules/?prefix=redcapcon2020&page=value\_lookup\_ajax&pid=230

$lookupUrl = $this->getUrl('value\_lookup\_ajax.php', false, false);

$lookupField = $this->getProjectSetting('lookup-field');

$setField = $this->getProjectSetting('set-field');

// need both confg settings for it to work!

if (empty($lookupField) || empty($setField)) { return; }

//write some JavaScript to the page that will perform the task we want

?>

<script type="text/javascript">

$(document).ready(function(){

// attach a click event handler to the button and show it

$('#DemoModuleLookup')

.on('click', function() {

// read the value entered in the lookup field

var entered = $('input[name=<?php echo $lookupField;?>]').val();

// send the entered value to the server and get back the max

$.post(

'<?php echo $lookupUrl;?>', // where to send the data

{ value\_entered: entered }, // the data to send (key-value pair object)

function( data ) { // the function to execute with the returned data

// write the result to the destination field

$('input[name=<?php echo $setField;?>]').val(data.result);

},

'json'

)

.fail(function() {

alert('value lookup failed');

});

return false;

});

});

</script>

<?php

}

Test it! Notice the POST on the Network tab of your browser developer tools.