# Displaying data from an SQLite database in a browser using datatables

This is the method that CHIELD uses. See the CHIELD Github page for source code.

The idea is that people access a webpage, and a javascript function fetches the data and displays it. The function calls a php function through an XMLHttpRequest or AJAX. The PHP script runs some SQL code on a database and returns the results.

I used datatables (see www.datatables.net) to display the records and allow searching. There is some custom code to allow each column to be searchable like in Glottlog list of languages. Actually, glottolog sends a new XMLHttpRequest for each character typed in the search box. This is useful if the database is very large and you don't want to load it into the page all at once.

## Administration

1) Set up an apache server on your mac so you can develop and test the website.

- Edit your apache cofig file to allow PHP: http://osxdaily.com/2012/09/10/enable-php-apache-mac-os-x/
- Start your apache server in a terminal:

```
> apachectl start
```

- Now, anything you put in the folder `/Library/WebServer/Documents/` will be accessible by going to a browser and going to `http://localhost/`.

2) Set up your website directory.

We have the following directory structure:

```
amazingdatabase/
    Site/
        index.html  (homepage)
        js/
            db.js
        php/
            getRecords.php
        browse.html
    data/
        db/
            amazingdatabase.sqlite
```

The folder "Site" is for things that will be publicly available. The folder "data" is for anything that is private and should not be directly accessible on the web.

You can now start building your website and testing it on your local machine. To put it on the web, you need to do step (3)

3) Upload the website to the excd virtual server. For EXCD, this is as follows:

Let's say your project is called AmazingDatabase, and you want it to be accessible from http://amazingdatabase.excd.org/

Send an email to service-desk@bristol.ac.uk to request:

- Your username to be added as an adimistrator on the excd virtual machine `arch-vm-1.arts.bris.ac.uk`.
- That a new public link is registered from http://amazingdatabase.excd.org/ to the folder /srv/amazingdatabase/Site/

The virtual can be accessed via ssh: arch-vm-1.arts.bris.ac.uk. The website data can be added to the folder `/srv/` (note initial slash - this not inside the home folder). You should add anything that can be accessed publicly to the folder e.g. `/srv/amazingdatabase/Site/` and anything that is prive to `/srv/amazingdatabase/data/`.

# Code

## Creating an sqlite database

The R package `RSQLite` can create an sqlite file from csv data. We can use `readr`'s `read_csv` to make sure the field types are being set correctly:

```
causal_links = read_csv("CausalLinks.csv", col_types = "cci")
variables = read_csv("Variables.csv", col_types = "cc")

my_db_file <- "amazingdatabase.sqlite"
my_db <- src_sqlite(my_db_file, create = TRUE)
my_db2 <- dbConnect(RSQLite::SQLite(), my_db_file)
dbWriteTable(my_db2, "causal_links",causal_links, overwrite=T)
dbWriteTable(my_db2, "variables",variables, overwrite=T)

dbDisconnect(my_db2)
```

## HTML

The file `browse.html`. The table heads must be specified correctly for the data that will be passed to it.

```
<!DOCTYPE html>
<html>
<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></s
cript>

<link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/v/bs4/dt-1
.10.16/sc-1.4.3/sl-1.2.4/datatables.min.css"/>
<script type="text/javascript" src="https://cdn.datatables.net/v/bs4/dt-1.10.16/sc
-1.4.3/sl-1.2.4/datatables.min.js"></script>

<script type="text/javascript" src="js/db.js"></script>

</head>
<body>
<table id="mytable" class="table table-striped table-bordered">
  <thead>
    <th>Authors</th>
    <th>Year</th>
    <th>Title</th>
    <th>Link</th>
  </thead>
  <tfoot>
    <th>Authors</th>
    <th>Year</th>
    <th>Title</th>
    <th>Link</th>
  </tfoot>
  <tbody>
  </tbody>
</table>
</body>
</html>
```

# Javascript

Prepare the datatable. The file `db.js`:

First, set up some global variables

```
var tableId = "mytable"; // datatable DOM id in html
var php_link = "php/getRecords.php";
// configuration of datatable:
var dtableConfig =  {
        ordering: true,
        lengthChange: false,
        order: [[0, "asc"]],
        paging: true,
        };
var dtable;
```

A function to prepare the datatable:

```
function preparePage(tableIdX,php_linkX){
    tableId = tableIdX;
    if(php_link!=""){
        php_link = php_linkX;
    }
    // set up column searching
    if(true){
        $('#'+tableId+' tfoot th').each( function () {
            var title = $(this).text();
            if(title=="Cor" || title=="Relation"){
                $(this).html( '<input type="text" placeholder="Search '+title+'" s
tyle="width:30px"/>' );
            } else{
                if(title=="Notes"){
                    $(this).html( '<input type="text" placeholder="Search '+title+
'" style="width:50px"/>' );
                } else{
                    $(this).html( '<input type="text" placeholder="Search '+title+
'" />' );
                    //$(this).html( '<input type="text" placeholder="Search" />' )
;
                }
            }
        } );
    }

}
```

## Function to request all records:

```
function requestLinks(php_link){
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
           // Typical action to be performed when the document is ready:
           updateLinksTable(xhttp.responseText);
        }
    };
    xhttp.open("GET", php_link, true);
    xhttp.send();
}
```

Function to recieve the records and put them into the datatable.

```
function updateLinksTable(text){

    var links = JSON.parse(text);
    // DataTable wants an array of arrays, so convert:
    var links2 = [];
    for(i in links){
        links2.push(Object.values(links[i]));
    }
    dtable_config = $.extend({data:links2},dtableConfig);
    dtable = $('#'+tableId).DataTable(dtable_config);
    addColumnSearching();
    }
```

## Function to request specific data

This function uses POST to pass a variable to a php script. The variable `params` should be a string in the form e.g. `"key=234372&name=someName"` ;

```
function requestRecord(php_link,params){

    var http = new XMLHttpRequest();
    http.open("POST", php_link, true);

    //Send the proper header information along with the request
    http.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

    http.onreadystatechange = function() {
    //Call a function when the state changes.
        if(http.readyState == 4 && http.status == 200) {
            updateLinksTable(http.responseText);
        }
    }
    http.send(params);
}
```

## Searching

Function to add search box for each column.

```
addColumnSearching(){
    // Add column searching
    dtable.columns().every( function () {
        var that = this;

        $( 'input', this.footer() ).on( 'keyup change', function () {
            if ( that.search() !== this.value ) {
                that
                    .search( this.value )
                    .draw();
            }
        } );
    } );
    // Move search box to header rather than footer
    $('#'+tableId+' tfoot tr').appendTo('#'+tableId+' thead');
    document.getElementById(tableId+'_filter').style.display = "none";
}
```

Finally, when the document loads, request the records:

```
$(document).ready(function(){
    preparePage(tableId,php_link);
    requestLinks(php_link);
}
```

# PHP

The file `Site/php/getRecords.php`. Note that the location of the sql file is relative to the Javascript file calling it

```
<?php

$pdo = new PDO('sqlite:../../data/db/amazingdatabase.sqlite');

// The SQL statement
$sql= "SELECT author, year, title, pk FROM documents";

$statement=$pdo->prepare($sql);
$statement->execute();
$results=$statement->fetchAll(PDO::FETCH_ASSOC);
$json=json_encode($results);
echo $json;

?>
```

Or filter the database. This should be called by `requestRecord()`, which passes a parameter 'key'. Note that binding the parameters is important to stop injection attacks:

```
<?php

$key = $_POST['key'];

$pdo = new PDO('sqlite:../../data/db/amazingdatabase.sqlite');

$sql= "SELECT record, citation FROM documents WHERE pk=:key";
$statement=$pdo->prepare($sql);
$statement->bindParam(':key', $key, PDO::PARAM_STR);
$statement->execute();
$results=$statement->fetchAll(PDO::FETCH_ASSOC);
$json=json_encode($results);
echo $json;

?>
```