**Building a Simple Address Book Application with jQuery and PHP**

Mambe Nanje Churchill | February 2010 | Open Source PHP Web Development

In this article by **Mambe Nanje Churchill**, we will explore the world of jQuery and PHP in achieving an AJAX style simple address book application using the JSON protocol. The basic application will help the user save, view and delete his or her contacts(name and phone). Its a basic application to demonstrate some key jQuery features like DOM transversing, jQquery event binding and AJAX calls with the $.ajax function.

Let's get along.

The application folder will be made up of five files:

1. *addressbook.css*
2. *addressbook.html*
3. *addressbook.php*
4. *addressbook.js*
5. *jquery.js*

*Addressbook.css* will contain the css for the interface styling, *addressbook.html* will contain the html source, *addressbook.js* contains  javascript codes, *addressbook.php* will mostly contain the server side code that will store the contacts to database, delete the contacts, provide updates and fetch the list of the contacts.

Let's look through the HTML

We include the scripts and the css file in the head tag in *addressbook.html* file.

<title>sample address book</title>   
<link rel="stylesheet" type="text/css" href="addressbook.css">   
<script type="text/javascript" src="jquery.js"></script>   
<script type="text/javascript" src="addressbook.js"></script>

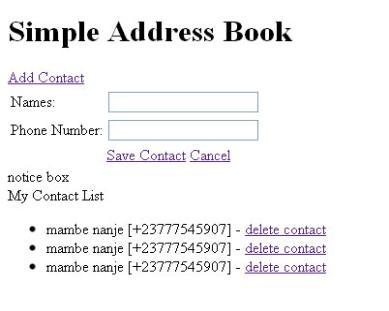
The code above includes the css for styling the application, jquery library for cross browser javascript and easy DOM access, and the *addressbook.js* contains functions that help user actions translated via javascript and ajax calls.

The *Body* tag should contain this:

<div id="Layer1">   
<h1>Simple Address Book</h1>   
<div id="addContact">   
              <a href="#add-contact" id="add-contact-btn">Add Contact</a>   
              <table id="add-contact-form">   
              <tr>   
              <td>Names:</td><td><input type="text" name="names" id="names"  /></td>   
              </tr>   
              <tr>   
              <td>Phone Number:</td><td><input type="text" name="phone" id="phone"  /></td>   
              </tr>   
              <tr>   
              <td>&nbsp;</td><td>   
              <a href="#save-contact" id="save-contact-btn">Save Contact</a>   
              <a href="#cancel" id="cancel-btn">Cancel</a>   
              </td>   
              </tr>   
              </table>   
</div>   
<div id="notice">   
              notice box   
</div>   
<div id="list-title">My Contact List</div>   
<ul id="contacts-lists">   
        <li>mambe nanje [+23777545907] - <a   
 href="#delete-id" class="deletebtn" contactid='1'> delete contact </a></li>   
        <li>mambe nanje [+23777545907] - <a   
 href="#delete-id" class="deletebtn" contactid='2'> delete contact</a></li>   
        <li>mambe nanje [+23777545907] - <a  
  href="#delete-id" class="deletebtn" contactid='3'> delete contact</a></li>   
</ul>   
</div>

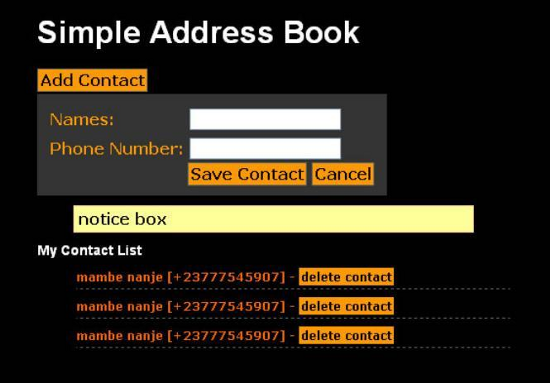
The above code creates an html form that provides input fields to insert new address book entries. It also displays a button to make it appear via javascript functions. It also creates a notification *div* and goes to display the contact list with delete button on each entry.

With the above code, the application will now look like this:



/\* CSS Document \*/   
    
body {   
              background-color: #000000;   
}   
#Layer1 {   
              margin:auto;   
              width:484px;   
              height:308px;   
              z-index:1;   
}   
#add-contact-form{   
              color:#FF9900;   
              font-weight:bold;   
              font-family:Verdana, Arial, Helvetica, sans-serif;   
              background-color:#333333;   
              margin-top:5px;   
              padding:10px;   
}   
#add-contact-btn{   
              background-color:#FF9900;   
              font-weight:bold;   
              font-family:Verdana, Arial, Helvetica, sans-serif;   
              border:1px solid #666666;   
              color:#000;   
              text-decoration:none;   
              padding:2px;   
              font-weight:bold;   
}   
#save-contact-btn{   
              background-color:#FF9900;   
              font-weight:bold;   
              font-family:Verdana, Arial, Helvetica, sans-serif;   
              border:1px solid #666666;   
              color:#000;   
              text-decoration:none;   
              padding:2px;   
              font-weight:bold;   
}   
#cancel-btn{   
              background-color:#FF9900;   
              font-weight:bold;   
              font-family:Verdana, Arial, Helvetica, sans-serif;   
              border:1px solid #666666;   
              color:#000;   
              text-decoration:none;   
              padding:2px;   
              font-weight:bold;   
}   
h1{   
              color:#FFFFFF;   
              font-family:Arial, Helvetica, sans-serif;   
}   
#list-title{   
              color:#FFFFFF;   
              font-weight:bold;   
              font-size:14px;   
              font-family:Arial, Helvetica, sans-serif;   
              margin-top:10px;   
}   
#contacts-lists{   
              color:#FF6600;   
              font-weight:bold;   
              font-family:Verdana, Arial, Helvetica, sans-serif;   
              font-size:12px;   
}   
#contacts-lists a{   
              background-color:#FF9900;   
              text-decoration:none;                
              padding:2px;   
              color:#000;   
              margin-bottom:2px;   
}   
#contacts-lists li{   
              list-style:none;   
              border-bottom:1px dashed #666666;   
              margin-bottom:10px;   
              padding-bottom:5px;   
}   
#notice{   
              width:400px;   
              margin:auto;   
              background-color:#FFFF99;   
              border:1px solid #FFCC99;   
              font-weight:bold;   
              font-family:verdana;   
              margin-top:10px;   
              padding:4px;   
}

The CSS code styles the HTML above and it ends up looking like this:



Now that we have our html and css perfectly working, we need to setup the database and the PHP server codes that will handle the AJAX requests from the jquery functions.

Create a MySQL database and executing the following SQL code will create the contacts table. This is the only table this application needs.

CREATE TABLE `contacts` (   
`id` INT NOT NULL AUTO\_INCREMENT PRIMARY KEY ,   
`names` VARCHAR( 200 ) NOT NULL ,   
`phone` VARCHAR( 100 ) NOT NULL   
);

Let's analyse the php codes. Remember, this code will be located in *addressbook.php*.

The database connection code

# FileName="Connection\_php\_mysql.htm"   
# Type="MYSQL"   
# HTTP="true"   
//configure the database paramaters   
$hostname\_packpub\_addressbook = "YOUR-DATABASE-HOST";   
$database\_packpub\_addressbook = "YOUR-DATABASE-NAME";   
$username\_packpub\_addressbook = "YOUR-DATABASE-USERNAME";   
$password\_packpub\_addressbook = "YOUR-DATABASE-PASSWORD";   
//connect to the database server   
$packpub\_addressbook = mysql\_pconnect($hostname\_packpub\_addressbook, $username\_packpub\_addressbook,  
  $password\_packpub\_addressbook) or trigger\_error(mysql\_error(),E\_USER\_ERROR);   
//selete the database   
mysql\_select\_db($database\_packpub\_addressbook);

the above code sets the parameters required for the database connection, then establishes a connection to the server and selects your database.

The PHP codes will then contain functions *SAVECONTACTS*, *DELETECONTACTS*, *GETCONTACTS*. These functions will do exactly as their name implies. Save the contact from AJAX call to the database, delete contact via AJAX request or get the contacts. The functions are as show below:

//function to save new contact   
/\*\*   
\* @param <string> $name //name of the contact   
\* @param <string> $phone //the telephone number of the contact   
\*/   
function saveContact($name,$phone){   
              $sql="INSERT INTO contacts (names , phone ) VALUES ('".$name."','".$phone."');";   
              $result=mysql\_query($sql)or die(mysql\_error());   
}   
//lets write a function to delete contact   
/\*\*   
\* @param <int> id //the contact id in database we wish to delete   
\*/   
function deleteContact($id){   
              $sql="DELETE FROM contacts where id=".$id;   
              $result=mysql\_query($sql);   
}   
    
//lets get all the contacts   
function getContacts(){   
              //execute the sql to get all the contacts in db   
              $sql="SELECT \* FROM contacts";   
              $result=mysql\_query($sql);   
              //store the contacts in an array of objects   
              $contacts=array();   
              while($record=mysql\_fetch\_object($result)){   
                            array\_push($contacts,$record);   
              }   
              //return the contacts   
              return $contacts;   
}

The codes above creates the functions but the functions are not called till the following code executes:

//lets handle the Ajax calls now   
$action=$\_POST['action'];   
//the action for now is either add or delete   
if($action=="add"){   
              //get the post variables for the new contact   
              $name=$\_POST['name'];   
              $phone=$\_POST['phone'];   
              //save the new contact   
              saveContact($name,$phone);   
              $output['msg']=$name." has been saved successfully";   
              //reload the contacts   
              $output['contacts']=getContacts();   
              echo json\_encode($output);   
}else if($action=="delete"){   
              //collect the id we wish to delete   
              $id=$\_POST['id'];   
              //delete the contact with that id   
              deleteContact($id);   
              $output['msg']="one entry has been deleted successfully";   
              //reload the contacts   
              $output['contacts']=getContacts();   
              echo json\_encode($output);   
}else{   
              $output['contacts']=getContacts();   
              $output['msg']="list of all contacts";   
              echo json\_encode($output);   
}

The above code is the heart of the *addressbook.php* codes. It gets the action from post variables sent via AJAX call in *addressbook.js* file, interprets the action and executes the appropriate function for either add, delete or nothing which will just get the list of contacts. *json\_encode()* function is used to encode the data in to Javascript Object Notation format; it will be easily interpreted by the javascript codes.

[**jQuery 1.3 with PHP**](http://www.packtpub.com/jquery-13-php/book/sr/mambe/0210?utm_source=sr_mambe_0210&utm_medium=content&utm_campaign=sanjay)

|  |  |
| --- | --- |
| [jQuery 1.3 with PHP](http://www.packtpub.com/jquery-13-php/book/sr/mambe/0210?utm_source=sr_mambe_0210&utm_medium=content&utm_campaign=sanjay) | Enhance your PHP applications by increasing their responsiveness through jQuery and its plugins. |

Published: October 2009   
eBook Price: $23.99  
Book Price: $39.99  
[See more](http://www.packtpub.com/jquery-13-php/book)

[Buy It Now](http://www.packtpub.com/jquery-13-php/book)

Now lets take a look at the javascript code found in *addressbook.js* file.

// JavaScript Document   
//global variable to hold client copy of the addresses in the database   
addresslist="";   
//function to ensure there are no duplicate entries   
function isDuplicate(names,phone){   
var isduplicate=false;   
for(var i=0;i<addresslist.length; i++){   
if(addresslist[i].names.toLowerCase()==names.toLowerCase()   
&& addresslist[i].phone.toLowerCase()==phone.toLowerCase()){   
isduplicate=true;   
}   
}   
return isduplicate;   
}   
//reload the address list using ajax   
function displayAddressList(items){   
//empty the contacts lists   
var list=$('#contacts-lists');   
//save a client copy of the items array for validation whenever its refreshed from server   
addresslist=items;   
//loop thru all the items and add to the list   
var lh="";   
for(var i=0;i<items.length; i++){   
lh+="<li>"+items[i].names;   
lh+=" [ "+items[i].phone+" ] ";   
lh+='<a href="#delete-id" class="deletebtn" contactid="'+items[i].id+'"> delete contact </a>'   
lh+="</li>";   
}   
list.html(lh);   
//set the delete button event after every reload   
setDeleteButtonEvents()   
}   
//function to set the save contact button event   
function setSaveButtonEvent(){   
$('#save-contact-btn').click(function(){   
//hide notice   
$('#notice').hide();   
//get the name and phone data   
var name=$('#names').val();   
var phone=$('#phone').val();   
//validate: ensure the name of phone is not empty, the name and phone not in dbase and   
//the name has only text and number has only numbers   
if(name=="" || phone==""){   
$('#notice').empty().html('the phone number or name field cannot be empty').show('slow');   
}else if(isDuplicate(name,phone)){   
$('#notice').empty().html('the contact info you specified is already in the database').show('slow');   
}else if(isNaN(new Number(phone))){   
$('#notice').empty().html('the phone field must contain valid numeric data').show('slow');   
}else if(name.match(/d/)){   
$('#notice').empty().html('the name field must not contain numeric input').show('slow');   
}else{   
//call the ajax save function   
$('#notice').empty().html('saving....').show();   
 $.ajax({   
 url: 'addressbook.php',   
 data: 'action=add&name='+name+'&phone='+phone,   
 dataType: 'json',   
 type: 'post',   
 success: function (j) {     
  //show the notice     
$('#notice').empty().html(j.msg);   
//empty the input fields   
$('#names').val('');   
$('#phone').val('');   
//refresh the address list   
displayAddressList(j.contacts);   
 }   
});   
}   
});   
}   
//function to set all delete button events   
function setDeleteButtonEvents(){   
$('.deletebtn').each(function(i){   
//set the delete event on each delete button   
$(this).click(function(){   
//confirm   
var answer = confirm("are you sure you want to delete ?");   
if(!answer){   
return;   
}   
//hide the form if its there   
$('#add-contact-form').hide();   
//set the delete notice   
$('#notice').empty().html('deleting...').show();   
//get the contactid of the current delete btn   
var id=$(this).attr('contactid');   
//call the ajax deleete function   
 $.ajax({   
 url: 'addressbook.php',   
 data: 'action=delete&id='+id,   
 dataType: 'json',   
 type: 'post',   
 success: function (j) {     
$('#notice').empty().html(j.msg);   
//refresh the address list   
displayAddressList(j.contacts);   
 }   
});   
});   
});   
}   
//initilize the javascript when the page is fully loaded   
$(document).ready(function(){   
//hide the add contact form   
$('#add-contact-form').hide();   
//hide the notice    
$('#notice').hide();   
//set the add contact form button event   
$('#add-contact-btn').click(function(){   
//hide the notice if its still there   
$('#notice').hide();   
//show the add contact form slowly when button is clicked   
$('#add-contact-form').show('slow');   
});   
//set the cancel button event   
$('#cancel-btn').click(function(){   
$('#add-contact-form').hide('slow');   
$('#notice').hide();   
//empty the input fields   
$('#names').val('');   
$('#phone').val('');   
});   
//set all the delete button events   
setDeleteButtonEvents();   
//set the save button event   
setSaveButtonEvent();   
//load the address list now   
//call the ajax save function   
 $.ajax({   
 url: 'addressbook.php',   
 data: '',   
 dataType: 'json',   
 type: 'post',   
 success: function (j) {   
//refresh the address list   
displayAddressList(j.contacts);   
},});   
});

The javascript contain the codes that make the HTML in *addressbook.html* communicate with the application user and the server side codes found in *addressbook.php*.

The above code initialises the application, hides the add contact form which is not visible by default and also hides the notification.  
It sets functions to click events for the *add contact* button.  
Lets analyse the functions:

1. *isDuplicate*: this function takes the name and phone fields, checks if it's duplicate, if so it returns true else returns false. It's used by the *SaveButtonEvent*, ensuring the system doesnt have duplicate entries.
2. *displayAddressList:* this function has codes that receive the JSON data from the server side, loops through and produces the live list of contacts from the database.
3. *setSaveButtonEvent:* this function when called adds and event to the Save Button which appears along with the add contacts form. This function ensures that when a user click on the save contact button, it validates the input ensuring its not empty, its not a duplicate, ensures the phone fields and names fields have only numeric and alpha data respectively, calls the addressbook.php with an action of add then, gets the results from php and displays the new list of contacts from JSON data and also the JSON data's message.
4. *SetDeleteButtonEvent:* this function adds an event to the Delete Button found next to every contact item in the list. Every contact listing, has its own delete button plus an extra html attribute contactid, this contactid gives us the contact we are about deleting at any given time this function is called. So we call addressbook.php via ajax and tells it the action is delete, we receive the respond from addressbook.php and set the new message and refresh the list of contacts, to ensure the deleted one no longer appears.

The final piece of code

//initilize the javascript when the page is fully loaded   
$(document).ready(function(){   
              //hide the add contact form                 
              $('#add-contact-form').hide();   
              //hide the notice   
              $('#notice').hide();   
              //set the add contact form button event   
              $('#add-contact-btn').click(function(){   
                            //hide the notice if its still there   
                            $('#notice').hide();   
                            //show the add contact form slowly when button is clicked   
                            $('#add-contact-form').show('slow');   
              });   
              //set the cancel button event   
              $('#cancel-btn').click(function(){   
                            $('#add-contact-form').hide('slow');   
                            $('#notice').hide();   
              });   
              //set all the delete button events   
              setDeleteButtonEvents();   
              //set the save button event   
              setSaveButtonEvent();   
              //load the address list now   
              //call the ajax save function   
                $.ajax({   
                              url: 'addressbook.php',   
                              data: '',   
                              dataType: 'json',   
                              type: 'post',   
                              success: function (j) {                                            
                                          //refresh the address list   
                                          displayAddressList(j.contacts);   
                            },});   
});

This code is the entry point to all functions in this application, and since this application is 100% javascript and AJAX using jQuery, without javascript it cant function. It involves no page refreshes, and works just like normal desktop applications with client server communciation using jQuery, JSON data and PHP.

When the javascript gets activated, the application looks like the live demo.

Live demo of the application:<http://www.afrovisiongroup.com/simpleaddressbook/>.

**Summary**

This tutorial explains how to go about a building a small address book application using jQuery, PHP and JSON protocol.

**If you have read this article you may be interested to view :**

* [How to Add Flair to your Actions with jQuery](http://www.packtpub.com/article/how-to-add-flair-to-actions-with-jquery?utm_source=sr_mambe_0210&utm_medium=content&utm_campaign=sanjay)
* [jQuery Table Manipulation](http://www.packtpub.com/article/jquery-table-manipulation-part1?utm_source=sr_mambe_0210&utm_medium=content&utm_campaign=sanjay)

[**jQuery 1.3 with PHP**](http://www.packtpub.com/jquery-13-php/book/sr/mambe/0210?utm_source=sr_mambe_0210&utm_medium=content&utm_campaign=sanjay)

|  |  |
| --- | --- |
| [jQuery 1.3 with PHP](http://www.packtpub.com/jquery-13-php/book/sr/mambe/0210?utm_source=sr_mambe_0210&utm_medium=content&utm_campaign=sanjay) | Enhance your PHP applications by increasing their responsiveness through jQuery and its plugins. |

Published: October 2009   
eBook Price: $23.99  
Book Price: $39.99  
[See more](http://www.packtpub.com/jquery-13-php/book)

[Buy It Now](http://www.packtpub.com/jquery-13-php/book)

**About the Author :**

**Mambe Nanje Churchill** is a Cameroonian software entrepreneur based in Buea. He is the founder and president of [AfroVisioN Group](http://www.afrovisiongroup.com/), a software company which provides domestic and outsourced solutions to a broad range of customers. When he is not writing software or doing some form of related business, you might find Churchill watching MTV Base or shooting and editing music videos.

Churchill is also the lead developer and President of technology of Show Ya Sef Network Africa [www.sysafrica.com](http://www.sysafrica.com/) where he designs and implements social media platforms like [naijaborn.com](http://naijaborn.com/) and [camerborn.com](http://camerborn.com/) . In his spare time, he likes to update his personal blog. Follow Churchill’s updates on Twitter.