Jquery Tutorial: Passing php arrays through JSON - Developer News

NEWS_PDF_AUTHOR: kaotik

NEWS_PDF_DATE: 2009/11/16 10:40:00

This tutorial is a continuation of my previous one. This will teach you the benefits of using JSON.

Method 2

What is JSON?
JSON stands for "javascript object notation", quote:
"Widely hailed as the successor to XML in the browser, JSON aspires to be nothing more than a simple, and elegant data format for the exchange of information between the browser and server; and in doing this simple task it will usher in the next version of the World Wide Web itself."
So basically think of JSON as being a conection language between PHP and Javascript (and in our case, jQuery). So when building web pages (with jquery) we can use ajax calls (be it $.load, $.ajax or other) to go from the client to the server, then we use JSON to go from the server back to the client. Now you might ask, couldn't we use json in both directions? Yes we could, but since we are using jquery, it already sends info nicely formated to the server in $_GET or $_POST format.

Step 1- Setting the php file

With json, setting the php file becomes much easier. Copy my previous tutorial into a new folder. Now open myserv.php and replace all code with this:

Notice that my arrays now have names "$list['name']" and I've also changed this:
$str=json_encode($list);
This line takes our php array and encodes it as a json string.
As I've said in my previous tutorial, javascript does not support associative arrays, however, we can use json to simulate them.

Step 2- The HTML

Open test.html and replace all code with this:

```
#ajaxBox { background-color:#FFFF99; border:thin solid #FF0000; width:70%; height:50px;}
#formHeader{text-align:center; font-size:18px; color:#0000FF;}
#myform{text-align:center;}
```

```
$(document).ready(function() { //Finish loading the entire page before processing any javascript

$("#hidden").hide();
$("#textfield").val("");
$("#textarea").val("");

$("#mylist a").bind("click", function() {
var hol=$(this).attr('myval');
var formContent ="action=getlink&link="+hol;

$.getJSON("myserv.php",formContent, function(json){
$("#textfield").val(json.name);
$("#textarea").val(json.desc);
$("#formHeader").text("Edit");
$("#ajaxBox").text("All info loaded OK");
}); //End json

}); //End click

}); //End ready function
```

cool site
new name
great space

Add New

Name
Desc

Now try running the page. Nice isn't it? Let me explain the changes. The piece of code that really matters here is this:

```
 $.getJSON("myserv.php",formContent, function(json){
$("#textfield").val(json.name);
$("#textarea").val(json.desc);
$("#formHeader").text("Edit");
$("#ajaxBox").text("All info loaded OK");
}); //End json
```

I'm now using a jquery function that does an ajax call and returns the data as a json object ($.getJSON). This data then gets placed in a variable called "json" which simulates an associative array. Notice this line:

$("#textfield").val(json.name);

I am assigning the form element "textfield" with "json.name". One of the cool things about json is, besides simulating associative arrays, you can write them as deep as you want, for ex:

```
 $list['country']='england';
$list['country']['city']='london';
$list['country']['city']['zip code']='12345';
etc
```

Now that we are retrieving properly formated data from the server, we can now generate html in a whole different way, but I'll leave that for a new tutorial.

Step 3- Looking back

So now we have looked at several ways of doing ajax calls and jquery function, let's briefly go over some of them.:

$.load - Does an ajax call and returns html.

$.getJSON - Does an ajax call and returns data formated as json.

$.change - Detect a change on a selector.

.bind("click", function()) - wait for a click on a selector

$.hide - hide a selector such as div, p, etc

$.show - show a selector

This tutorial is a continuation of my previous one. This will teach you the benefits of using JSON.

Method 2

What is JSON?
JSON stands for "javascript object notation", quote:
"Widely hailed as the successor to XML in the browser, JSON aspires to be nothing more than a simple, and elegant data format for the exchange of information between the browser and server; and in doing this simple task it will usher in the next version of the World Wide Web itself."
So basically think of JSON as being a conection language between PHP and Javascript (and in our case, jQuery). So when building web pages (with jquery) we can use ajax calls (be it $.load, $.ajax or other) to go from the client to the server, then we use JSON to go from the server back to the client. Now you might ask, couldn't we use json in both directions? Yes we could, but since we are using jquery, it already sends info nicely formated to the server in $_GET or $_POST format.

Step 1- Setting the php file

With json, setting the php file becomes much easier. Copy my previous tutorial into a new folder. Now open myserv.php and replace all code with this:

Notice that my arrays now have names "$list['name']" and I've also changed this:
$str=json_encode($list);
This line takes our php array and encodes it as a json string.
As I've said in my previous tutorial, javascript does not support associative arrays, however, we can use json to simulate them.

Step 2- The HTML

Open test.html and replace all code with this:

```
#ajaxBox { background-color:#FFFF99; border:thin solid #FF0000; width:70%; height:50px;}
#formHeader{text-align:center; font-size:18px; color:#0000FF;}
#myform{text-align:center;}
```

```
$(document).ready(function() { //Finish loading the entire page before processing any javascript

$("#hidden").hide();
$("#textfield").val("");
$("#textarea").val("");

$("#mylist a").bind("click", function() {
var hol=$(this).attr('myval');
var formContent ="action=getlink&link="+hol;

$.getJSON("myserv.php",formContent, function(json){
$("#textfield").val(json.name);
$("#textarea").val(json.desc);
$("#formHeader").text("Edit");
$("#ajaxBox").text("All info loaded OK");
}); //End json

}); //End click

}); //End ready function
```

cool site
new name
great space

Add New

Name
Desc

Now try running the page. Nice isn't it? Let me explain the changes. The piece of code that really matters here is this:

```
 $.getJSON("myserv.php",formContent, function(json){
$("#textfield").val(json.name);
$("#textarea").val(json.desc);
$("#formHeader").text("Edit");
$("#ajaxBox").text("All info loaded OK");
}); //End json
```

I'm now using a jquery function that does an ajax call and returns the data as a json object ($.getJSON). This data then gets placed in a variable called "json" which simulates an associative array. Notice this line:

$("#textfield").val(json.name);

I am assigning the form element "textfield" with "json.name". One of the cool things about json is, besides simulating associative arrays, you can write them as deep as you want, for ex:

```
 $list['country']='england';
$list['country']['city']='london';
$list['country']['city']['zip code']='12345';
etc
```

Now that we are retrieving properly formated data from the server, we can now generate html in a whole different way, but I'll leave that for a new tutorial.

Step 3- Looking back

So now we have looked at several ways of doing ajax calls and jquery function, let's briefly go over some of them.:

$.load - Does an ajax call and returns html.

$.getJSON - Does an ajax call and returns data formated as json.

$.change - Detect a change on a selector.

.bind("click", function()) - wait for a click on a selector

$.hide - hide a selector such as div, p, etc

$.show - show a selector