Combining PHP, JSON and jQuery Screencast

By Quentin Zervaas, 14 June 2010, tagged in Ajax, jQuery, JSON, PHP

Videos

- http://www.youtube.com/watch?v=C71adS_dD-E

Transcript

[0:06] In this screencast we're going to create a form with a dynamic dropdown. The dropdown will show a list of cities that we retrieve from a JSON datasource.

[0:17] Our first step is to create the script that sends the JSON data. We load up our editor. I've already created a few blank scripts here but we'll fill these in as we go.

[0:30] So `json.php`. We create an array. This is just a PHP array with a list of cities. Adelaide, Perth, Melbourne Sydney. And we can send back this in JSON format - which is basically just JavaScript code - using `json_encode()`. We just pass the array and save the script.

[1:10] Now, back in Chrome we load the file (`index.php`). That's the array we just created. If you actually look in developer tools, the script is here under resources. It's sent it with content type `text/html`. Technically this isn't really a problem. It will work, but to be correct it needs to be `application/json` content type.

[1:37] So just back in `json.php` use the `header()` function, `content-type` and then save, reload. Now if we look here, it's `application/json` and Chrome gives us a warning here, that's just because we tried to display it as a normal HTML page even though it's not.

Listing 1. The JSON data source (json.php)

```
<?php
    $cities = array(
        'Adelaide',
        'Perth',
        'Melbourne',
        'Sydney'
    );

    header('Content-type: application/json');
    echo json_encode($cities);
?>
```

[2:08] That's the data source done. Next we're going to create some JavaScript to actually output that. Let's go back here. The script we'll just update now to load a select box. Now this doesn't actually do anything, all we're going to do is populate this. So let's call it `city`, and give it an id of `#city-list`, so we can refer to it later in our JavaScript. And we'll just put a blank element that instructs the user to select a city. And we save that, then go back to the `index.php` script.

[3:01] Now we've got this dropdown which we'll populate shortly. We're using jQuery in this. To make things a little bit simpler we're going to use Google's hosted version of jQuery. We can search Google jQuery and I think it's the first link. So we just click on jQuery and it shows you the versions that they host. You can either use their loader or just go straight to the path. We'll get the minified version (`jquery.min.js`) because it loads quicker. We don't need the full source code.

[3:43] So let's just load that here. So that's all we need to do to import jQuery into our script, and we're about to create this `script.js` file, so we'll just load that one as well.

Listing 2. The main index file that displays the form (index.php)

```
<html>
    <head>
        <title>PHP + jQuery Example - PhpRiot.com</title>
        <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></
script>
        <script type="text/javascript" src="script.js"></script>
    </head>
    <body>
        <div>
            <h1>PHP + jQuery Example - PhpRiot.com</h1>

            <select name="city" id="city-list">
                <option value="">Select a city...</option>
            </select>
        </div>
    </body>
</html>
```

Copyright 2010 Quentin Zervaas, All Rights Reserved

Combining PHP, JSON and jQuery Screencast

By Quentin Zervaas, 14 June 2010, tagged in Ajax, jQuery, JSON, PHP

[4:15] Okay so that's now loading, it doesn't actually do anything yet, but what we can do is just ensure that jQuery is loaded. So this is the "on dom loaded" event. Save that one and go back to the browser. We don't need this any more, now refresh. Hopefully we'll get an alert box just to ensure that jQuery is loaded, and there we go. That means we can keep going.

[4:53] Now, we go back, we can remove this alert. Now what we want to do is use the jQuery `getJSON()` function. So for that, we call `$.getJSON()`. The script is called `json.php`, so put that in, and the callback is the second argument, and when the callback is called the data returned from the script is passed as the first argument.

[5:32] So that's our placeholder. You can have a look at that before we do anything with it. Just throw in a `console.log()`, pass the data, save and return the Chrome. Now if you load developer tools. Let's refresh this page. So now you can see this script here after the JavaScript is loaded it performed the JSON query. So that's what we created earlier on but you can this output that we wrote to the console. So we're most of the way there, all we need to do now is use this data to populate the select box.

[6:20] So returning to our editor. So we just call `$.each()`, pass the array as the first argument and the callback as the second. The first argument to the callback is the index, so `0`, `1`, `2`, `3` in this case, and the second argument is the value, which is the name of the city, which is what we're going to use.

[6:50] For each iteration of this loop we're going to write a new option to the select box. So first we need to get the select box. So we're going to call it `var select`, and we called that `#city-list`, and then in each iteration option, create the new `option`, so we set the `value` attribute, so when the form is submitted the value is passed in, and we set the HTML using the `.html()` function, so we just pass the city name again. And finally we append it to the select, so `appendTo(select)`.

Listing 3. JavaScript to populate the select box with JSON (script.js)

```
jQuery(function($) {
        $.getJSON('json.php', function(json) {
                var select = $('#city-list');

                $.each(json, function(k, v) {
                        var option = $('<option />');

                        option.attr('value', v)
                              .html(v)
                              .appendTo(select);
                });
        });
});
```

[7:40] Now we can save, return to Chrome, and refresh. Now we've got our developer tools still open so you can see they're all appearing here, and in the dropdown. And that's all there is to it! Now we haven't actually done anything with this list, but you can use this inside your normal forms, or you can trigger the load after a particular btuton is pressed. Here we've just made it simple and doing it onload, but you can really do it however you need to.

Copyright 2010 Quentin Zervaas, All Rights Reserved