Create your WEB Application with MVC paradigm and OPP – PAP support

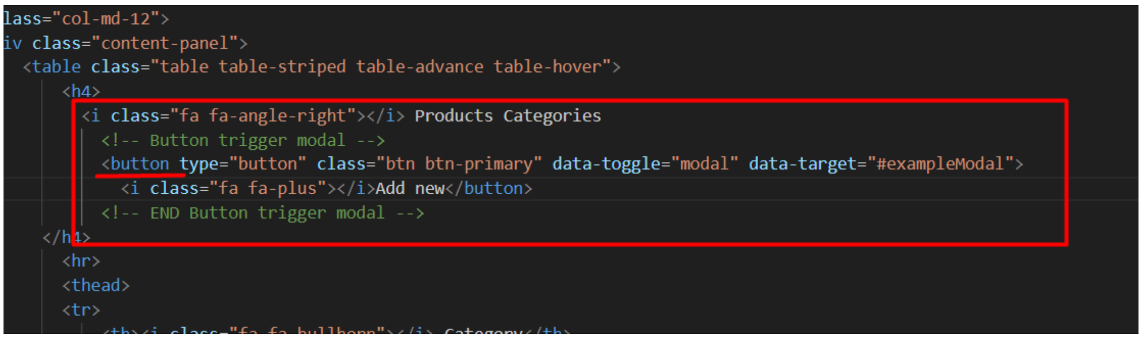
Tutorial 15

Okay, in this tutorial, you will prepare your template to be ready to receive AJAX functions. At this moment, you should know how AJAX works. If you are not familiar with the concepts, please go to Google and do some research.

In my example, I will use these concepts, but feel free to use the traditional way. You have already created in your menu the links to your CRUD views, with ADD, EDIT, or DELETE.

Okay, you also need to prepare your template. Follow my example. I will use modals from the Bootstrap library. These tools permit using the same page and controller to manipulate your data forms.

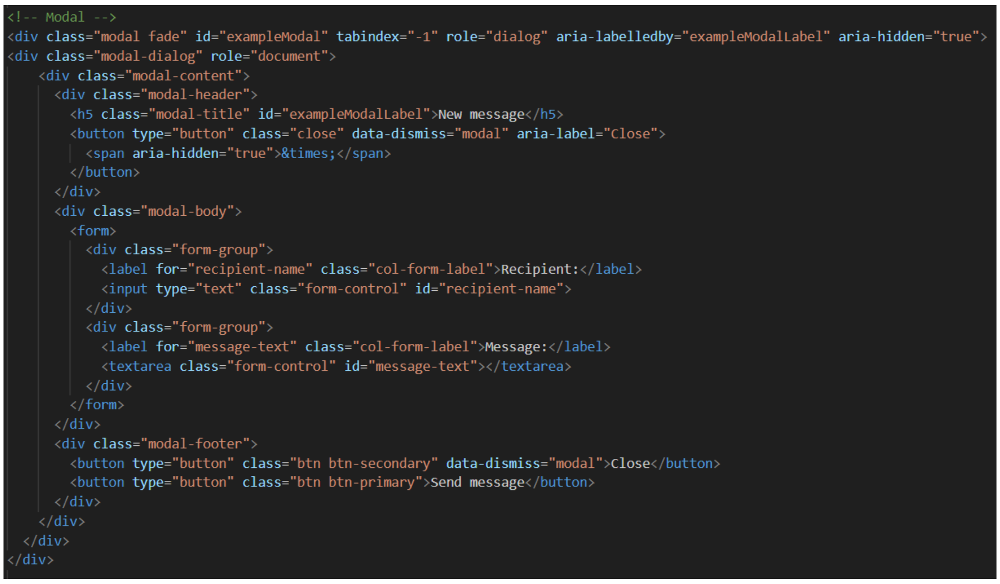
Please go to the Bootstrap website and search for the modal tool, then adapt it to your code, as shown in the example below:



```
lass="col-md-12">
iv class="content-panel">
  <table class="table table-striped table-advance table-hover">
      <h4>
        <i class="fa fa-angle-right"></i> Products Categories
          <!-- Button trigger modal -->
          <button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal">
            <i class="fa fa-plus"></i>Add new</button>
          <!-- END Button trigger modal -->
    </h4>
      <hr>
      <thead>
      <tr>
        <th><i class="fa fa-bullhorn"></i> Category</th>
```

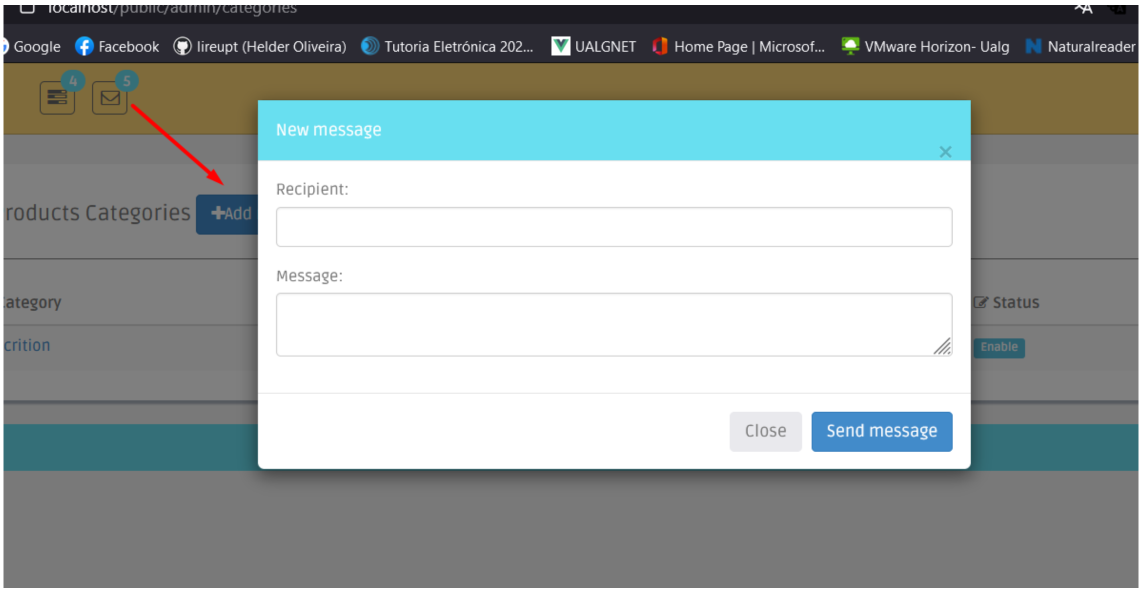
In these images you have the button example, that will call the modal.

Next introduce the modal code:



```html
<!-- Modal -->
<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
<div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">New message</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <form>
          <div class="form-group">
            <label for="recipient-name" class="col-form-label">Recipient:</label>
            <input type="text" class="form-control" id="recipient-name">
          </div>
          <div class="form-group">
            <label for="message-text" class="col-form-label">Message:</label>
            <textarea class="form-control" id="message-text"></textarea>
          </div>
        </form>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Send message</button>
      </div>
    </div>
  </div>
</div>
```

And the final result when I click in the button is:

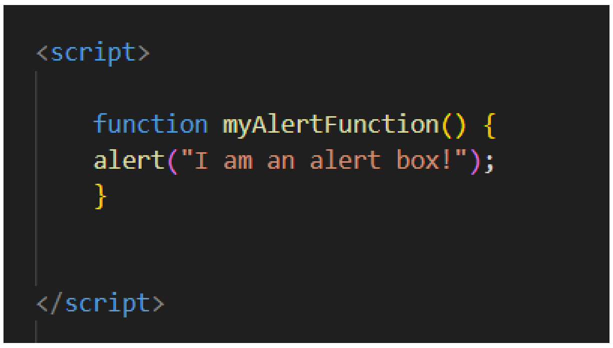


I chose a modal with buttons and a form inside to facilitate my design work, but as I said before, feel free to add your own code.

So, when I click on the button, the background stays faded, and the box appears in front of the page. This technique permits having multiple forms on one simple page and controller, instead of multiple pages if you do it the traditional way.

Now let’s prepare your send data form to add a new category.

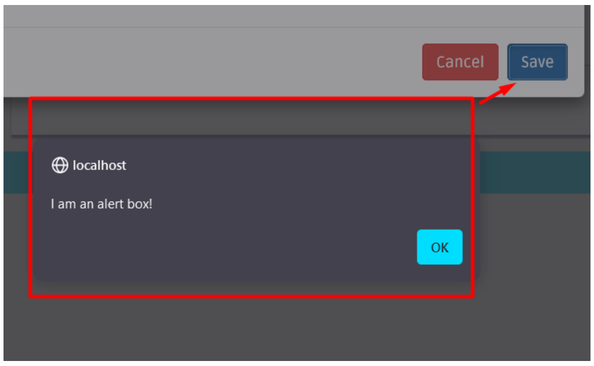
To do that, you need to create a AJAX function inside your view code, above the html code inside a script tags as the example:



```
<script>

    function myAlertFunction() {
    alert("I am an alert box!");
    }


</script>
```

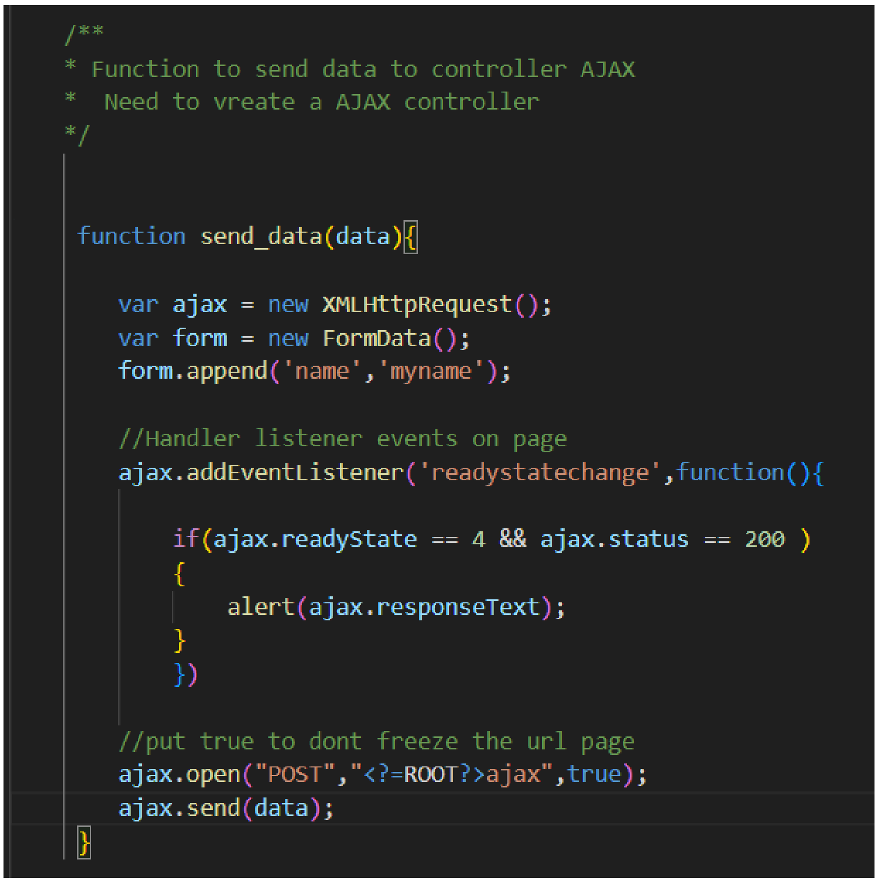
First you will test your click event function, to test the save button state. Please put the code above to test the event click.

The result should be like this when you made the save click:



I am an alert box!

Next create a function that will handle with your data form:



```
    /**
    * Function to send data to controller AJAX
    *  Need to vreate a AJAX controller
    */


    function send_data(data){

       var ajax = new XMLHttpRequest();
       var form = new FormData();
       form.append('name','myname');

       //Handler listener events on page
       ajax.addEventListener('readystatechange',function(){

           if(ajax.readyState == 4 && ajax.status == 200 )
           {
               alert(ajax.responseText);
           }
           })

       //put true to dont freeze the url page
       ajax.open("POST","<?=ROOT?>ajax",true);
       ajax.send(data);
    }
```

Okay, the send_data function is designed to send data to a server using AJAX. It starts by creating instances of XMLHttpRequest and FormData. A key-value pair (name, myname) is added to the FormData object.

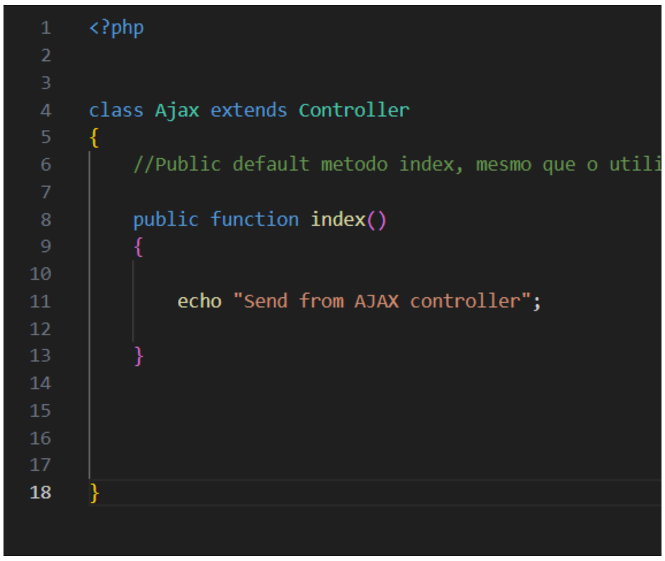
An event listener is then set up to monitor changes in the request's state.

If the request completes successfully (indicated by a readyState of 4 and a status of 200), an alert displays the server's response text.

The function then opens a POST request to a specified URL asynchronously (so it doesn't freeze the page) and sends the provided data to the server. This enables data submission and response handling without reloading the page.

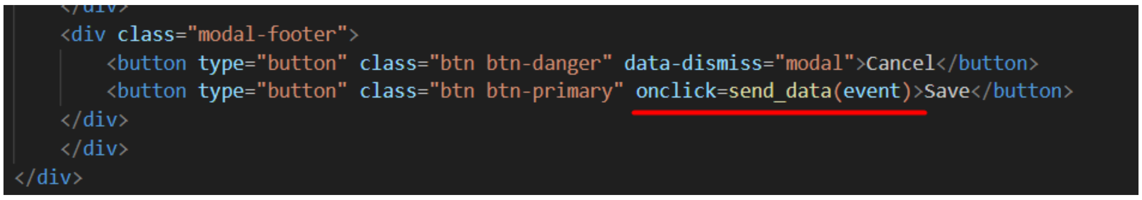
Now create a new controller inside the folder controller and name it as ajax.

Next put a echo message inside the index first function like the example:



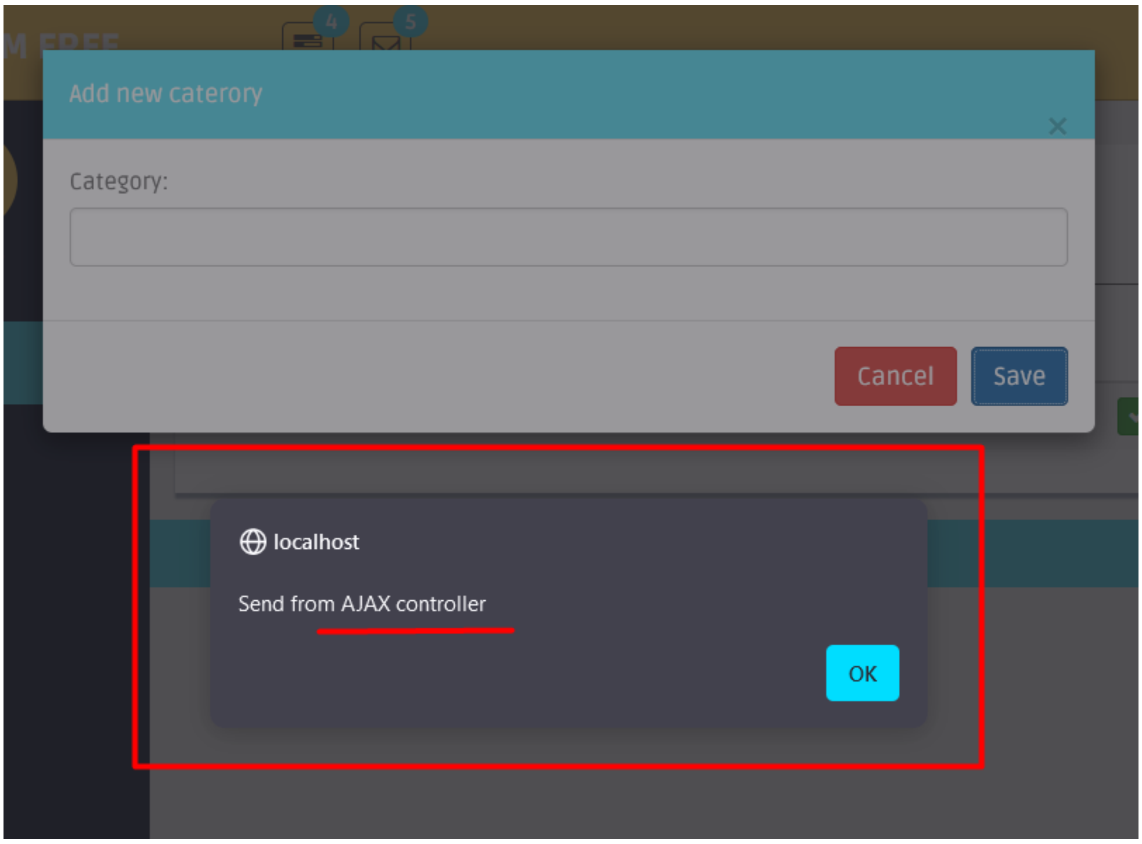
```php
<?php


class Ajax extends Controller
{
    //Public default metodo index, mesmo que o utili

    public function index()
    {

        echo "Send from AJAX controller";

    }




}
```

Now add a onclick event to your button to call the function:



```html
    <div class="modal-footer">
        <button type="button" class="btn btn-danger" data-dismiss="modal">Cancel</button>
        <button type="button" class="btn btn-primary" onclick=send_data(event)>Save</button>
    </div>
    </div>
</div>
```

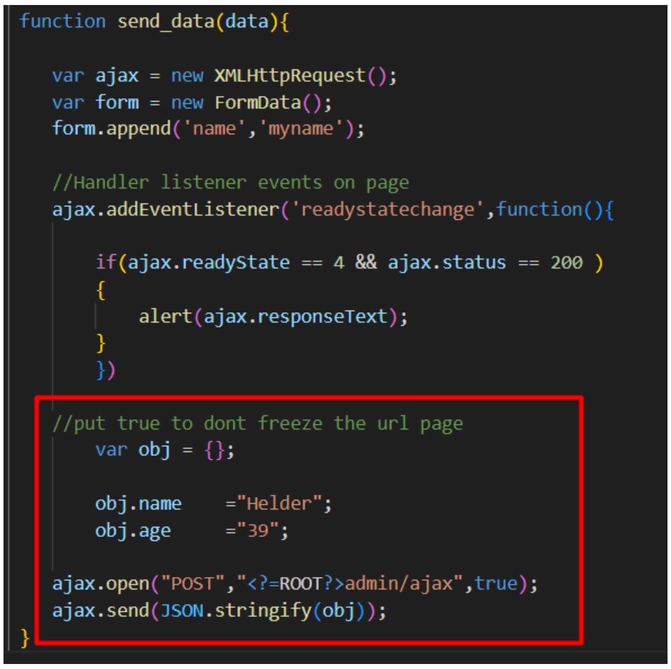
At the end, test the "Save" button. If everything is working correctly, the result should match the image below when you click "Save":



Send from AJAX controller

At the end, test the "Save" button. If everything is working correctly, the result should match the image above when you click "Save".
Okay, now our function is work correctly, let's get some data to send to your controller.

First let's do some tests. Let's change our function send_data().

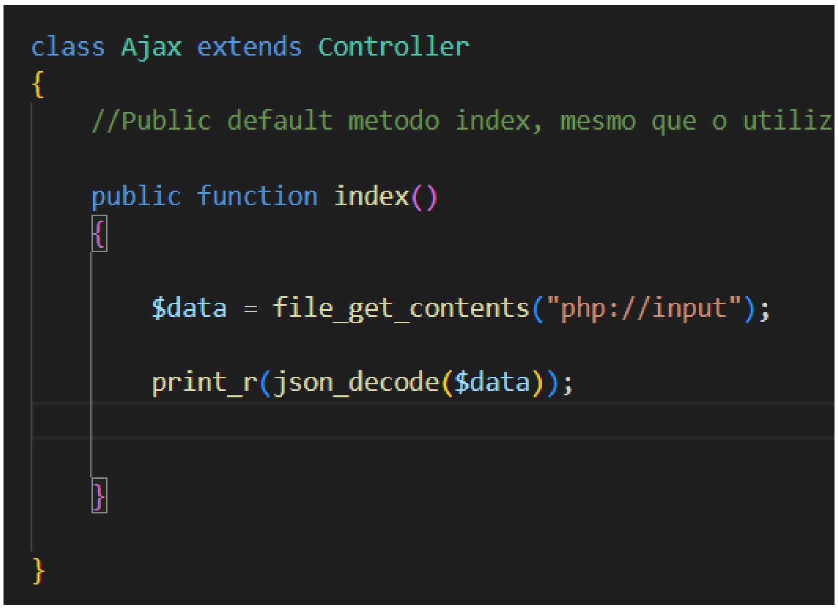


```
function send_data(data){

   var ajax = new XMLHttpRequest();
   var form = new FormData();
   form.append('name','myname');

   //Handler listener events on page
   ajax.addEventListener('readystatechange',function(){

       if(ajax.readyState == 4 && ajax.status == 200 )
       {
           alert(ajax.responseText);
       }
       })

   //put true to dont freeze the url page
       var obj = {};

       obj.name    ="Helder";
       obj.age     ="39";

   ajax.open("POST","<?=ROOT?>admin/ajax",true);
   ajax.send(JSON.stringify(obj));
}
```

We will use JSON.stringify to serialize your data, to handle encoding and decoding.

If you are not familiar with this concepts, please make some research before advance in tutorial.

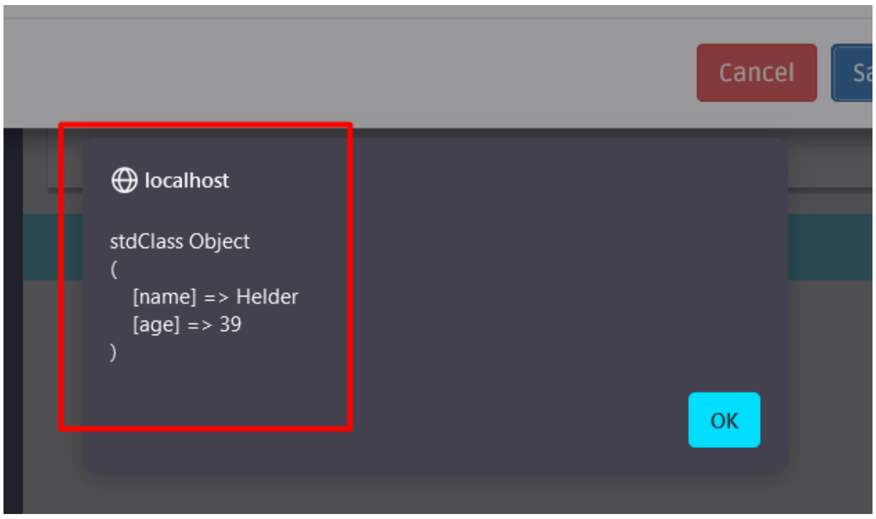
Ok, now change your controller:



```
class Ajax extends Controller
{
    //Public default metodo index, mesmo que o utiliz

    public function index()
    {

        $data = file_get_contents("php://input");

        print_r(json_decode($data));


    }

}
```

The function call file_get_contents("php://input") in PHP is used to read raw data from the request body. Next you use the json decode to align your data inside an object.

Ok, now test again you save button, the result sould be like the image bellow:

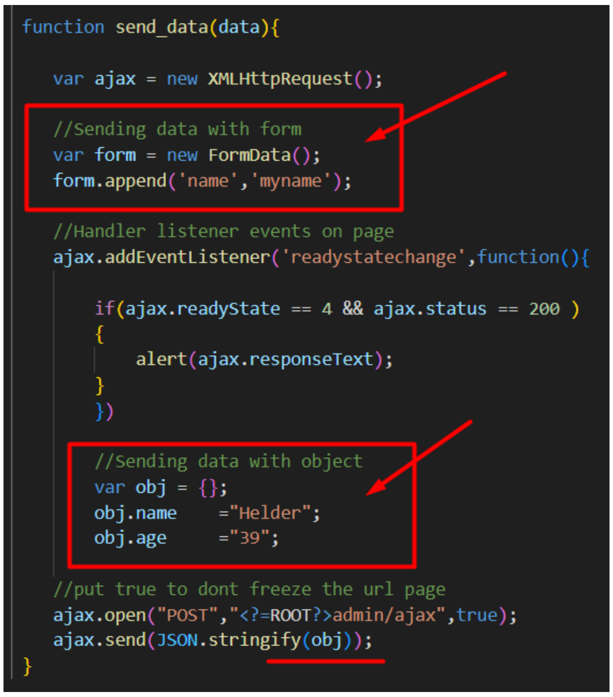


```
stdClass Object
(
    [name] => Helder
    [age] => 39
)
```

Now you have your data serialized inside an object.

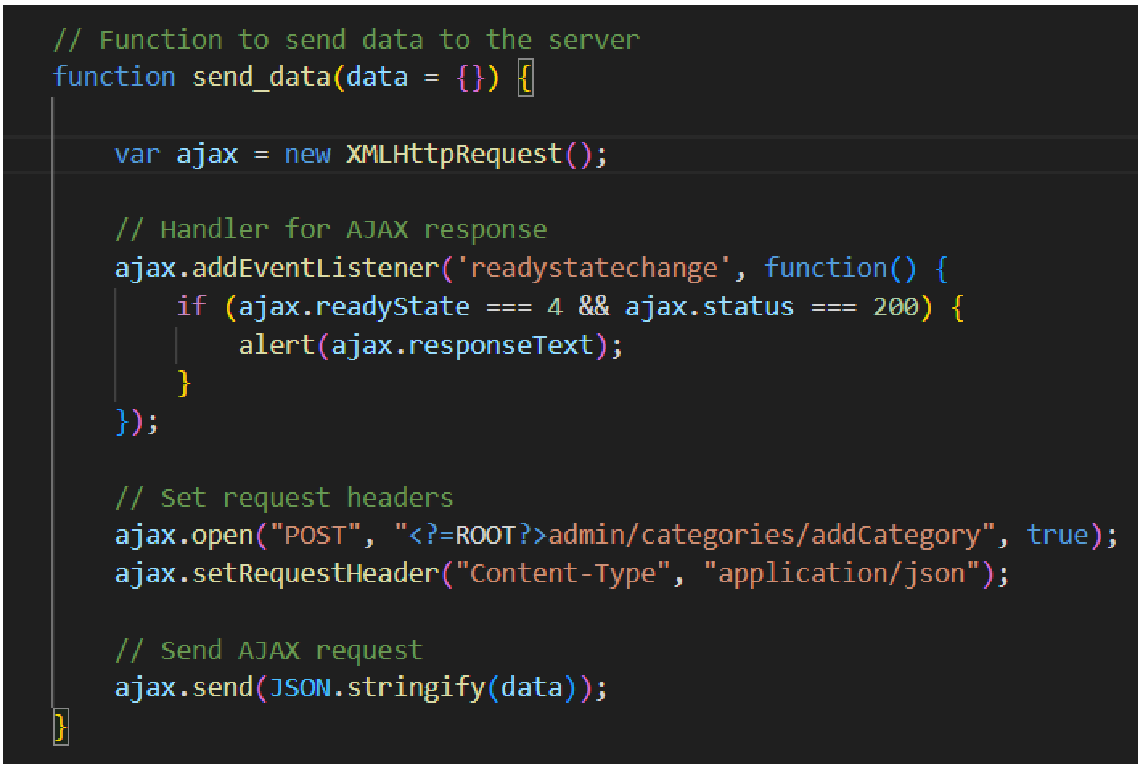
Let’s continuous.

Ok, if you see well your function, you may send data from two aways, using a form like the first example, or using a pais key->value object, as the second example:



```
function send_data(data){

   var ajax = new XMLHttpRequest();

   //Sending data with form
   var form = new FormData();
   form.append('name','myname');

   //Handler listener events on page
   ajax.addEventListener('readystatechange',function(){

       if(ajax.readyState == 4 && ajax.status == 200 )
       {
           alert(ajax.responseText);
       }
       })

       //Sending data with object
       var obj = {};
       obj.name     ="Helder";
       obj.age      ="39";

   //put true to dont freeze the url page
   ajax.open("POST","<?=ROOT?>admin/ajax",true);
   ajax.send(JSON.stringify(obj));
}
```

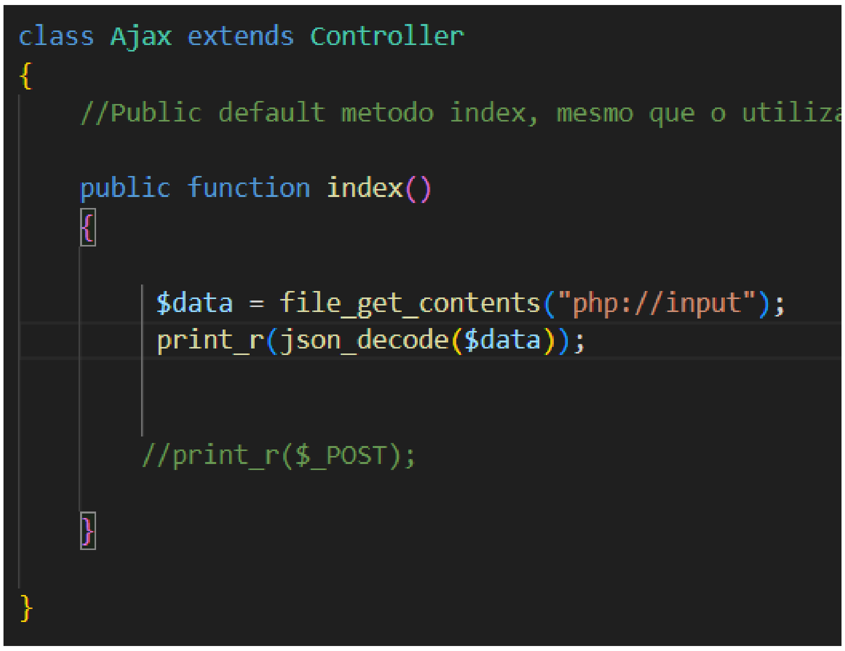
I will use the JSON encode example, with a POST method. Please change again your function:



```
// Function to send data to the server
function send_data(data = {}) {

    var ajax = new XMLHttpRequest();

    // Handler for AJAX response
    ajax.addEventListener('readystatechange', function() {
        if (ajax.readyState === 4 && ajax.status === 200) {
            alert(ajax.responseText);
        }
    });

    // Set request headers
    ajax.open("POST", "<?=ROOT?>admin/categories/addCategory", true);
    ajax.setRequestHeader("Content-Type", "application/json");

    // Send AJAX request
    ajax.send(JSON.stringify(data));
}
```

Pay attention to you path on arrow sign, because, when the page reloads, need to know where it is going next.

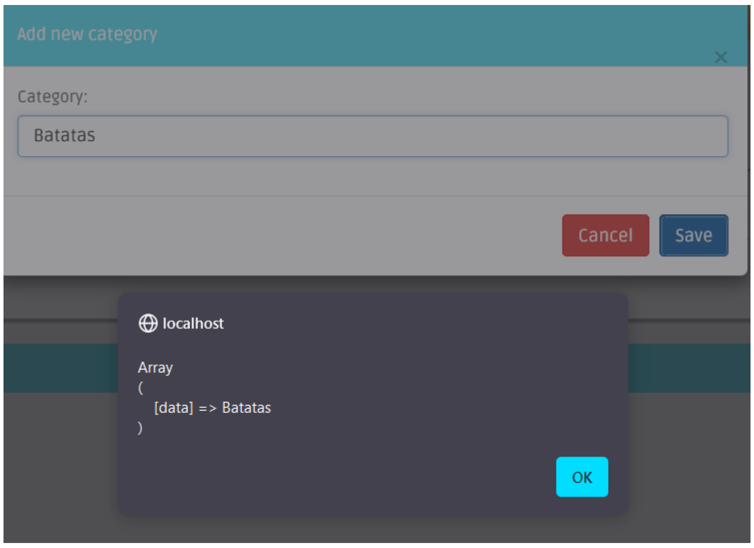
Now inside your controller change also to this code:



```php
class Ajax extends Controller
{
    //Public default metodo index, mesmo que o utiliza

    public function index()
    {

        $data = file_get_contents("php://input");
        print_r(json_decode($data));


        //print_r($_POST);

    }

}
```

Okay, you can use the bothe example to collect your data inside your controller. I will use the file_get_content() method.

Result:



```
Array
(
    [data] => Batatas
)
```

Remember, always comment your code, to facilitate in the future to read them.

END Tutorial 15

Reference: **PHP Ecommerce website development | Create Categories View | MVC OOP - Quick programming**

Template: NICEADMIN, design by BOOTSTRAP

Modal: https://getbootstrap.com/docs/4.0/components/modal/