

## Laravel - Session

Sessions are used to store information about the user across the requests. Laravel provides various drivers like **file**, **cookie**, **apc**, **array**, **Memcached**, **Redis**, and **database** to handle session data. By default, file driver is used because it is lightweight. Session can be configured in the file stored at **config/session.php**.

### Accessing Session Data

To access the session data, we need an instance of session which can be accessed via HTTP request. After getting the instance, we can use the **get()** method, which will take one argument, **key**, to get the session data.

```
$value = $request->session()->get('key');
```

You can use **all()** method to get all session data instead of **get()** method.

### Advertisement

### Storing Session Data

Data can be stored in session using the **put()** method. The **put()** method will take two arguments, the **key** and the **value**.

```
$request->session()->put('key', 'value');
```

### Deleting Session Data

The **forget()** method is used to delete an item from the session. This method will take **key** as the argument.

```
$request->session()->forget('key');
```

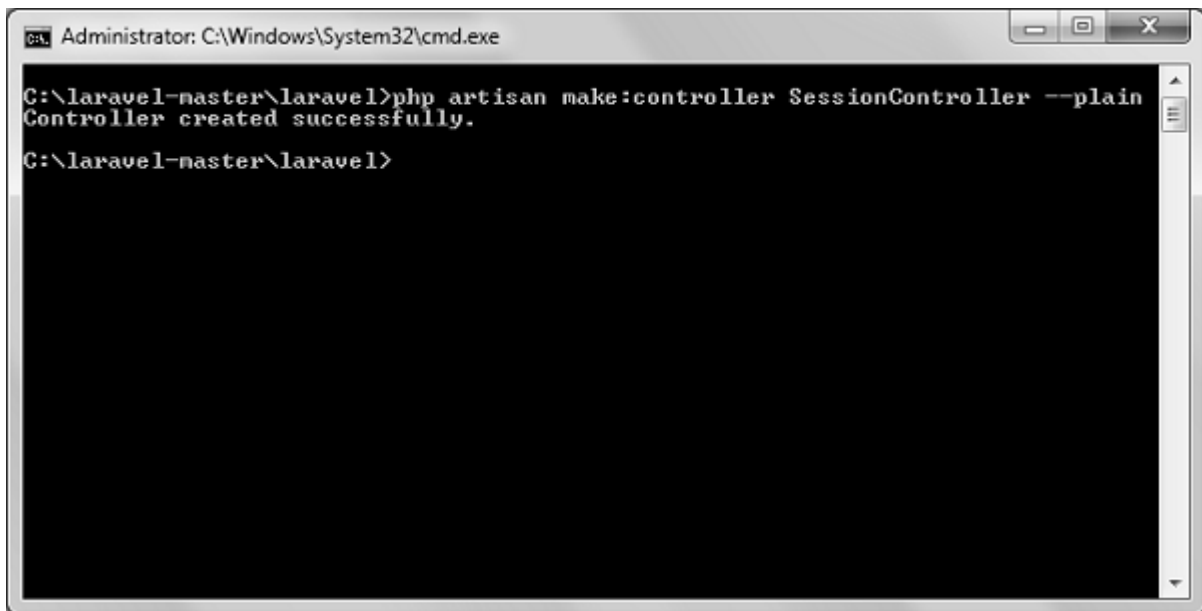
Use **flush()** method instead of **forget()** method to delete all session data. Use the **pull()** method to retrieve data from session and delete it afterwards. The **pull()** method will also take **key** as the argument. The difference between the **forget()** and the **pull()** method is that **forget()** method will not return the value of the session and **pull()** method will return it and delete that value from session.

### Example

**Step 1** – Create a controller called **SessionController** by executing the following command.

```
php artisan make:controller SessionController --plain
```

**Step 2** – After successful execution, you will receive the following output –

A screenshot of a Windows command prompt window titled "Administrator: C:\Windows\System32\cmd.exe". The window shows the following text: 

```
C:\laravel-master\laravel>php artisan make:controller SessionController --plain
Controller created successfully.
C:\laravel-master\laravel>
```

**Step 3** – Copy the following code in a file at

**app/Http/Controllers/SessionController.php.**

**app/Http/Controllers/SessionController.php**

**<?php**

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Http\Requests;

use App\Http\Controllers\Controller;

class SessionController extends Controller {

public function accessSessionData(Request \$request) {

if(\$request->session()->has('my\_name'))

echo \$request->session()->get('my\_name');

else

echo 'No data in the session';

}

public function storeSessionData(Request \$request) {

```
$request->session()->put('my_name','Virat Gandhi');

echo "Data has been added to session";

}

public function deleteSessionData(Request $request) {

    $request->session()->forget('my_name');

    echo "Data has been removed from session.";

}

}
```

**Step 4** – Add the following lines at **app/Http/routes.php** file.

**app/Http/routes.php**

```
Route::get('session/get','SessionController@accessSessionData');

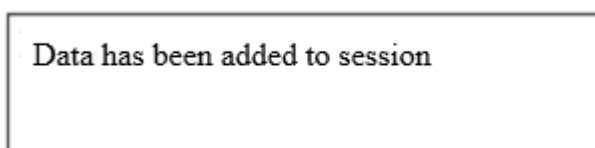
Route::get('session/set','SessionController@storeSessionData');

Route::get('session/remove','SessionController@deleteSessionData');
```

**Step 5** – Visit the following URL to **set data in session**.

<http://localhost:8000/session/set>

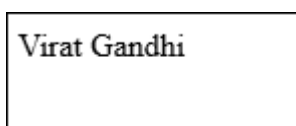
**Step 6** – The output will appear as shown in the following image.

A screenshot of a web browser window displaying the text "Data has been added to session" in a simple, black, sans-serif font. The text is centered within a white rectangular area that has a thin black border.

**Step 7** – Visit the following URL to **get data from session**.

<http://localhost:8000/session/get>

**Step 8** – The output will appear as shown in the following image.

A screenshot of a web browser window displaying the text "Virat Gandhi" in a simple, black, sans-serif font. The text is centered within a white rectangular area that has a thin black border.

**Step 9** – Visit the following URL to **remove session data**.

<http://localhost:8000/session/remove>

## Laravel - File Uploading

Uploading Files in Laravel is very easy. All we need to do is to create a view file where a user can select a file to be uploaded and a controller where uploaded files will be processed.

In a view file, we need to generate a file input by adding the following line of code.

```
Form::file('file_name');
```

In `Form::open()`, we need to add **files=>true** as shown below. This facilitates the form to be uploaded in multiple parts.

```
Form::open(array('url' => '/uploadfile','files'=>'true'));
```

Example

**Step 1** – Create a view file called **resources/views/uploadfile.php** and copy the following code in that file.

**resources/views/uploadfile.php**

```
<html>

<body>

    <?php

        echo Form::open(array('url' => '/uploadfile','files'=>'true'));

        echo 'Select the file to upload.';

        echo Form::file('image');

        echo Form::submit('Upload File');

        echo Form::close();

    ?>

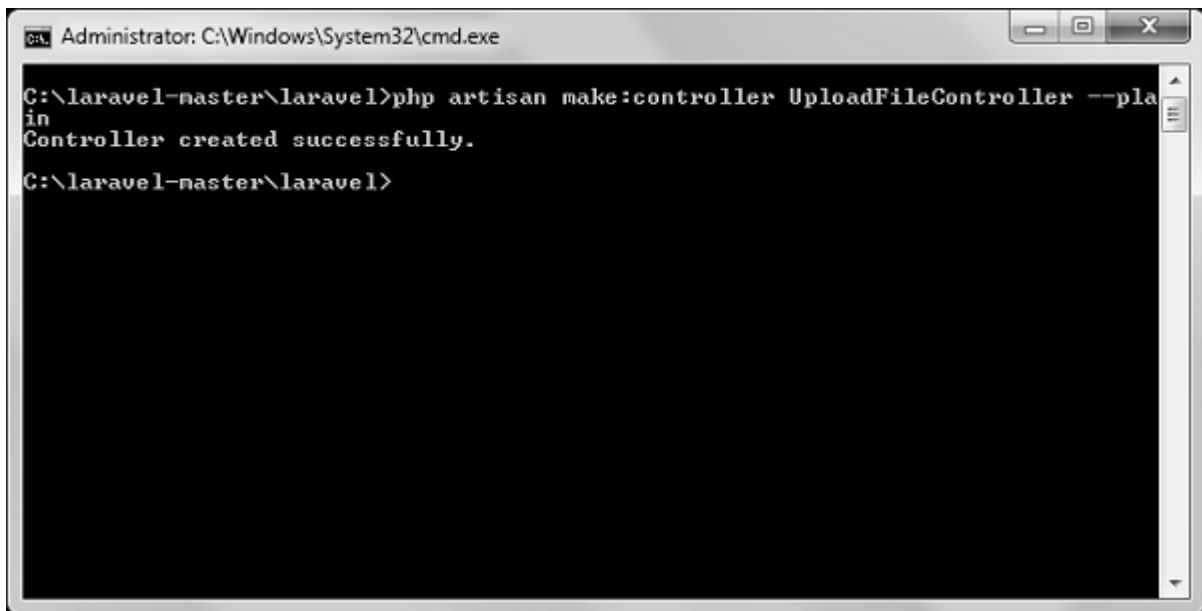
</body>

</html>
```

**Step 2** – Create a controller called **UploadFileController** by executing the following command.

```
php artisan make:controller UploadFileController --plain
```

**Step 3** – After successful execution, you will receive the following output –



```
Administrator: C:\Windows\System32\cmd.exe
C:\laravel-master\laravel>php artisan make:controller UploadFileController --plain
Controller created successfully.
C:\laravel-master\laravel>
```

**Step 4** – Copy the following code in

**app/Http/Controllers/UploadFileController.php** file.

**app/Http/Controllers/UploadFileController.php**

**<?php**

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Http\Requests;

use App\Http\Controllers\Controller;

class UploadFileController extends Controller {

public function index() {

return view('uploadfile');

}

public function showUploadFile(Request \$request) {

\$file = \$request->file('image');

```
//Display File Name
```

```
echo 'File Name: '.$file->getClientOriginalName();
```

```
echo '<br>';
```

```
//Display File Extension
```

```
echo 'File Extension: '.$file->getClientOriginalExtension();
```

```
echo '<br>';
```

```
//Display File Real Path
```

```
echo 'File Real Path: '.$file->getRealPath();
```

```
echo '<br>';
```

```
//Display File Size
```

```
echo 'File Size: '.$file->getSize();
```

```
echo '<br>';
```

```
//Display File Mime Type
```

```
echo 'File Mime Type: '.$file->getMimeType();
```

```
//Move Uploaded File
```

```
$destinationPath = 'uploads';
```

```
$file->move($destinationPath,$file->getClientOriginalName());
```

```
}
```

```
}
```

**Step 5** – Add the following lines in **app/Http/routes.php**.

## app/Http/routes.php

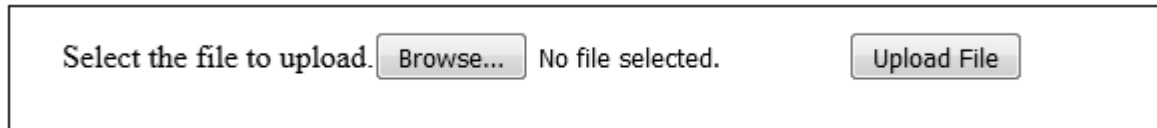
```
Route::get('/uploadfile','UploadFileController@index');
```

```
Route::post('/uploadfile','UploadFileController@showUploadFile');
```

**Step 6** – Visit the following URL to test the upload file functionality.

<http://localhost:8000/uploadfile>

**Step 7** – You will receive a prompt as shown in the following image.



**Ajax (Asynchronous JavaScript and XML)** is a set of web development techniques utilizing many web technologies used on the client-side to create asynchronous Web applications. Import jquery library in your view file to use ajax functions of jquery which will be used to send and receive data using ajax from the server. On the server side you can use the response() function to send response to client and to send response in JSON format you can chain the response function with json() function.

json() function syntax

```
json(string|array $data = array(), int $status = 200, array $headers = array(), int $options)
```

Example

**Step 1** – Create a view file called **resources/views/message.php** and copy the following code in that file.

```
<html>
```

```
<head>
```

```
<title>Ajax Example</title>
```

```
<script src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
```

```
</script>
```

```
<script>
```

```
function getMessage() {
```

```

$.ajax({

    type:'POST',

    url:'/getmsg',

    data:'_token = <?php echo csrf_token() ?>',

    success:function(data) {

        $("#msg").html(data.msg);

    }

});

}

</script>

</head>

<body>

    <div id = 'msg'>This message will be replaced using Ajax.

    Click the button to replace the message.</div>

    <?php

        echo Form::button('Replace Message',['onClick'=>'getMessage()']);

    ?>

</body>

</html>

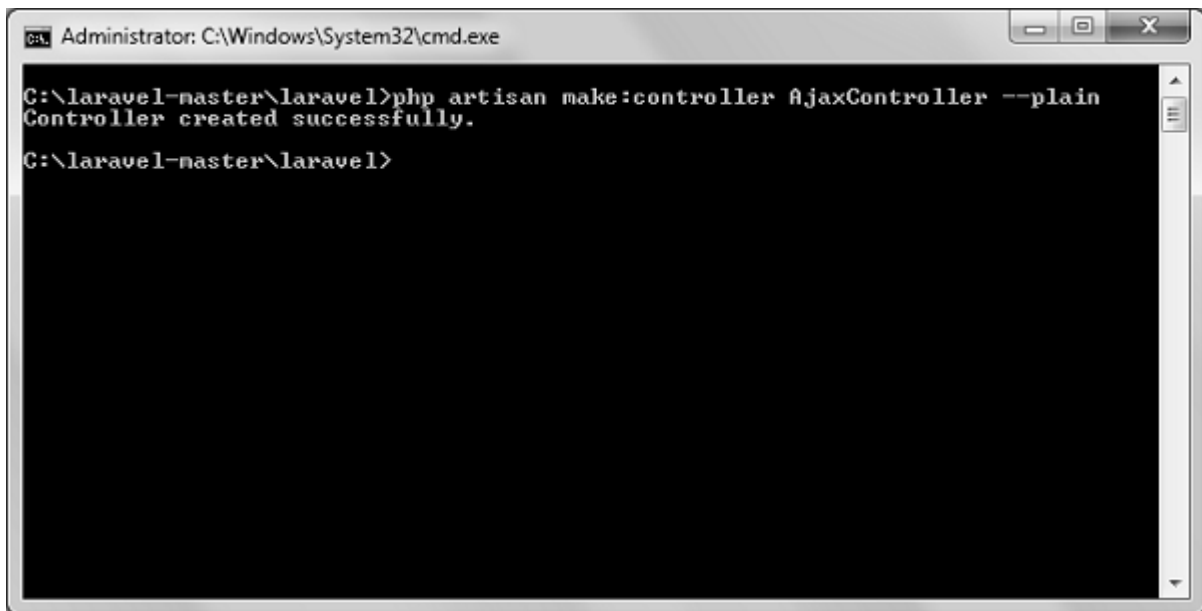
```

**Step 2** – Create a controller called **AjaxController** by executing the following command.

```
php artisan make:controller AjaxController --plain
```

**Step 3** – After successful execution, you will receive the following output –





```
Administrator: C:\Windows\System32\cmd.exe
C:\laravel-master\laravel>php artisan make:controller AjaxController --plain
Controller created successfully.
C:\laravel-master\laravel>
```

**Step 4** – Copy the following code in

**app/Http/Controllers/AjaxController.php** file.

**app/Http/Controllers/AjaxController.php**

**<?php**

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Http\Requests;

use App\Http\Controllers\Controller;

class AjaxController extends Controller {

public function index() {

    \$msg = "This is a simple message.";

    return response()->json(array('msg'=> \$msg), 200);

}

}

**Step 5** – Add the following lines in **app/Http/routes.php**.

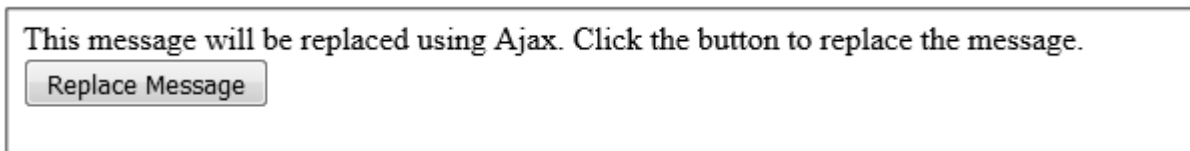
## app/Http/routes.php

```
Route::get('ajax',function() {  
  
    return view('message');  
  
});  
  
Route::post('/getmsg','AjaxController@index');
```

**Step 6** – Visit the following URL to test the Ajax functionality.

<http://localhost:8000/ajax>

**Step 7** – You will be redirected to a page where you will see a message as shown in the following image.



**Step 8** – The output will appear as shown in the following image after clicking the button.

