

# SERVER-SIDE WEB PROGRAMMING UNIT5: STORING INFORMATION WITH DATABASES

# Index

2

- Working with files
  - Semidynamic pages
  - Handling file uploads
  - PDF extensions
  - Pagination

# 1. Semidynamic Pages

3

- Database driven pages use a great deal more processing time on the computer that runs the web server software than do plain old HTML files.
- How often does front information actually change? Once an hour? Once a day? Would your site really suffer if changes took effect after a some delay?
- By converting high-traffic dynamic pages into semidynamic equivalents you can significantly improve on your web server's performance.

# 1. Semidynamic Pages

- Say that you have a controller script (`index.php`) that uses a PHP template to generate your front page, which provides a summary of new content on your site.
- You'll probably find that this is one of the most requested pages on your site.
- Instead of using a controller script to handle every request for the front page of your site, you can use the PHP code to generate a static snapshot of the PHP template's output and put this snapshot online as **`index.html`**.

# 1. Semidynamic Pages

5

- In order to implement this, you will have to learn how to manage files.
- Important filesystem methods:
  - *file\_get\_contents*
  - *file\_put\_contents*
  - *file\_exists*
  - *copy*
  - *unlink*

# 1. Semidynamic Pages

6

## □ Exercise 1:

1. Take the last version of your *bookstore* project.
2. Create a *generate.php* file inside *bookstore* /core folder and add the following code adapting the corresponding paths to the ones of your project:

# 1. Semidynamic Pages

7

```
1 <?php
2 $srcurl = 'http://dwes.local/lesson3/Part31/bookstore/public/';
3 $tempfilename = $_SERVER['DOCUMENT_ROOT'] .
4     '/lesson3/Part31/bookstore/public/tempindex.html';
5 $targetfilename = $_SERVER['DOCUMENT_ROOT'] .
6     '/lesson3/Part31/bookstore/public/index.html';
7
8 if (file_exists($tempfilename))
9 {
10     unlink($tempfilename);
11 }
12
13 $html = file_get_contents($srcurl);
14 if (!$html)
15 {
16     $error = "Unable to load $srcurl. Static page update aborted!";
17     echo $error;
18     exit();
19 }
20
21 if (!file_put_contents($tempfilename, $html))
22 {
23     $error = "Unable to write $tempfilename. Static page update aborted!";
24     echo $error;
25     exit();
26 }
27
28 copy($tempfilename, $targetfilename);
29 unlink($tempfilename);
30 echo "Home page generated succesfully!";
```

generate.php

# 1. Semidynamic Pages

8

4. Call *generate.php* in your browser:

```
localhost/lesson3/Through_MVC/bookstore_WithSemi/core/generate.php
```

5. Now, if you check inside the public folder of the project a default *index.html* should have been created!



## 2. Handling File Uploads

9

- We are going to change our form, in order to add also a file as the cover of the corresponding book.

- Exercise 2:

1. In your bookstore DB, follow these steps:

- A. Create this new table called *files*:

#	Nombre	Tipo
1	id 📄	int(11)
2	filename	varchar(255)
3	mimetype	varchar(50)
4	filedata	mediumblob
5	books_id 📄	int(11)
6	updated_at	timestamp
7	created_at	timestamp

- B. Make *id* from *files* primary key from that table.

**Note:** In this example, we are using the same DB. However, the better would be files to be saved in a different DB. Moreover, if files are too heavy, the best way would be to save in DB only the name of the file.

## 2. Handling File Uploads

10

2. Now that you have the new table created, you will have to add a new model class called Files, the same way you did with Books:

```
1  <?php
2
3  use Illuminate\Database\Eloquent\Model as Eloquent;
4
5  class Files extends Eloquent
6  {
7      |
8  ★ public function books()
9      {
10         |         return $this->belongsTo('Books');
11         |     }
12     }
13
```

- Have a look at line 8, where you are creating a method called *books()*, in order to define which is the relation between Books model and Files model.
- Launch the “composer update” command to update the autoloading files.

## 2. Handling File Uploads

11

- We need also to change Books model in order to finish the definition of the relation of the two models. Add this *files()* method inside of Books model:

```
11     public function files()  
12     {  
13         return $this->hasOne('Files');  
14     }  
15 }  
16
```

## 2. Handling File Uploads

12

3. Now you are going to change *form.html.php* in order to upload files:

- ▣ To have the file itself submitted with the form, we need to add `enctype="multipart/form-data"` to the `<form>` tag:

```
<form action="upload.php" method="post" enctype="multipart/form-data" c  
<fieldset>
```



- ▣ Also we need to add an input file type:

```
<label for="cover">Select file to upload:  
<input type="file" id="cover" name="cover"></label><br />
```



## 2. Handling File Uploads

13

4. The last thing will be to change `addBook()` from the `Books` model method in order to insert previous information and the uploaded file correctly:

a) Prepare the information given from the server:

```
32         if (!is_uploaded_file($_FILES['cover']['tmp_name'])) {  
33             $error = 'There was no file uploaded!';  
34             Messages::setMsg($error, 'error');  
35             return;  
36         }  
37         $uploadfile = $_FILES['cover']['tmp_name'];  
38         $uploadname = $_FILES['cover']['name'];  
39         $uploadtype = $_FILES['cover']['type'];  
40         $uploaddata = file_get_contents($uploadfile);
```

b) After having the information in variables, **on your own** make the corresponding insert query into filestore table!!!!

Steps:

1. Create a Books object and save it (what we already had).
2. Create a Files object and save it **related** to recently created Books object ...why?? Because the books id must be saved into files table!! In order to do so... use the corresponding method!



## 2. Handling File Uploads

14

- Now that we have cover photographs uploaded, we are going to modify *listBooks* in order to show inside the table each books cover.
- Exercise 3:
  1. Check if you get the files table information whenever you call to index method inside the controller. Check the content of the books you get from the database... you should get everything: not only the list of books, also the files related to each books.
  2. Add a new column in *books.html.php* in order the image to be displayed when the page is loaded.
  3. See how could you print the image into the recently created column...

[Add your own book](#)

Here are all the books in the database:

Cover	Name	Author	Price	ISBN	Publisher	Published date	
	1	1	1	1	1	1	<a href="#">Details</a>
	x	x	x	x	x	0	<a href="#">Details</a>

# 3. PDF extensions

15

- PHP has several libraries for generating PDF documents.
- We are going to use FPDF library.

# 3. PDF extensions

16

□ Example: Before continuing with our *bookstore*, try this example in order to get familiar with FPDF library

□ Check out and play with this methods:

- *setFont*
- *setTextColor*
- *cell*
- *ln*

```
$pdf = new FPDF('P', 'in', 'Letter');  
$pdf->addPage();  
  
$pdf->setFont('Arial', 'B', 24);  
$pdf->cell(0, 0, "Top Left!", 0, 0, 'L');  
  
$pdf->setFont("Arial", 'U', 12);  
$pdf->setTextColor(128, 128, 128);  
$pdf->cell(0, 0, "Top Right!", 0, 0, 'R');  
  
$pdf->ln(4.5);  
$pdf->setFont("Arial", 'IBU', 20);  
$pdf->setTextColor(255, 0, 0);  
$pdf->cell(0, 0, "This is the middle!", 0, 0, 'C');  
  
$pdf->ln(5.3);  
$pdf->setFont("Times", 'IU', 15);  
$pdf->cell(0, 0, "Bottom Left!", 0, 0, 'L');  
$pdf->cell(0, 0, "Bottom Right!", 0, 0, 'R');  
  
$pdf->output();
```



# 3. PDF extensions

17

- This time we are going to add a link into *book.html.php* in order it to generate a PDF with the information contained in a table.
- Exercise 4:
  1. Firstly, you are going to add a link that makes a call to the controller with a parameter called *pdf* (in order to call to a *pdf* method from the *BooksController*).



Mario

k

2

k

[Export to PDF](#)



ID	Name	Sex	Authors	ISBN	Publisher	Published date
56	El castillo	23	Franz Kafka	1234	OMM	0
57	q	q	q	q	q	0
59	n	n	n	n	n	1
65	f	f	f	f	f	0
71	m	m	m	m	m	0
73	b	b	b	b	b	0

# 3. PDF extensions

18

2. We are going to make use of FPDF library. In order to use it go into your terminal and make use of composer to load that library into your project (vendor folder):


```
composer require setasign/fpdf
```

# 3. PDF extensions

19

3. Apart from that, you are going to create a new php file in your project called *PDF.php* inside core folder:

- It creates a PDF class that extends FPDF.
- PDF class contains a method called *BuildTable*.
- The code is, initially, given for you...



```
class PDF extends FPDF
{
    function BuildTable($header,$data)
    {
        //Colors, line width and bold font
        $this->SetFillColor(230,230,230);
        $this->SetTextColor(255);
        $this->SetDrawColor(128,0,0);
        $this->SetLineWidth(.3);
        $this->SetFont('', 'B');
        //Header
        // make an array for the column widths
        $w=array(12,40,12,40);
        // send the headers to the PDF document
        for($i=0;$i<count($header);$i++)
            $this->Cell($w[$i],7,$header[$i],1,0,'C',1);
        $this->Ln();
        //Color and font restoration
        $this->SetFillColor(175);
        $this->SetTextColor(0);
        $this->SetFont('');
        //now spool out the data from the $data array
        $fill=0; // used to alternate row color backgrounds
        foreach($data as $row)
        {
            $this->Cell($w[0],6,$row['id'],'LR',0,'C',$fill);
            $this->Cell($w[1],6,$row['name'],'LR',0,'C',$fill);
            $this->Cell($w[2],6,$row['price'],'LR',0,'C',$fill);
            $this->Cell($w[3],6,$row['authors'],'LR',0,'C',$fill);
            $this->Ln();
            $fill = ! $fill;
        }
        $this->Cell(array_sum($w),0,'','T');
    }
}
```

# 3. PDF extensions

20

4. Run to load of the class we have just created:

```
MacBook-Pro-de-Ines:bookstore_WithPDF Ines$ composer dump-autoload  
Generating autoload files
```

# 3. PDF extensions

21

5. Now, inside the BooksController, you will have to create a new method called pdf:

```
public function pdf(){  
    $books=Books::all();  
    // start and build the PDF document  
    ★ $pdf = new PDF();  
    //Column titles  
    $header=array('ID','Name','Price', 'Authors');  
    $pdf->SetFont('Arial','',14);  
    $pdf->AddPage();  
    ★ // call the table creation method  
    $pdf->BuildTable($header,$books);  
    $pdf->Output('ListBooks.pdf','D');  
}
```

- With this code you are going to create the file, call to BuildTable method from PDF class and make the file directly downloaded.

# 3. PDF extensions

22

6. Test it, you should already be generating the PDF file.

Once the previous steps work, you will have to change the given code, in order to print all the columns, not just four...



ID	Name	Price	Authors	ISBN	Publisher	Published date
56	El castillo	23	Franz Kafka	1234	OMM	0
57						0

# 4. Pagination

23



Add some pagination (5 per page, for instance) into our books/index controller method and view (index.html.php).