

Forum:[Hack Forums](#)

Thread: [HTML/JS Templating/Mass Spinning - PHP panel?](#)

+mK - December 20, 2022, 4:41 pm

I really have no clue how to word this. I've looked through Google, but I think my keywords are just dog shit. I don't know what to call it.

I have a single HTML file that I'm going to use as my base template. I want to be able to specify 40 or so specific div classes, image sources, references to titles in a .JS file, color profiles in my .CSS file. My problem is, is that I want to do make 500 or so of HTML files out of that original single one, with all varying content. Is this possible? I was thinking about somehow making a PHP panel to render HTML based on the classes or contents of rendered elements. I also went down a Vue/Handlebars/Mustache rabbit hole, but I'm not entirely sure if that's what I want either. Maybe I just didn't see good enough examples.

Example of what I'm trying to do (play pretend with me for a second):

My single HTML page "template": <https://themexriver.com/appilo-theme/app-showcase-dark/>

If I wanted to grab the top 100 mobile apps on the market and create their own custom landing page, with the template perfectly matching the single app in question, how would I do that easier?

End goal: Have a shit ton of input fields/vars tied to content in the HTML and be able to just fill those in, in like 30 seconds, I'll have completed one landing page.

I appreciate any advice!

Romeos - December 20, 2022, 8:24 pm

this is extremely confusing tbh.

Shellcode™ - December 20, 2022, 11:24 pm

Dynamic content (HTML, CSS, JS) is absolutely doable just by using Raw PHP. Also you will not really find "examples" that do EXACTLY what you have written here anywhere, but instead separate scripts for every moving piece - And you should properly stitch/adjust them together.

This is something very doable as I said earlier, but if you are not experienced developer, I strongly suggest you to hire someone.

Hackers - December 21, 2022, 4:59 am

Yes, very easy to do with:

- JS
- PHP

You could make it random on page load, or use PHP to generate a template & save it as an HTML file.

For speed, Python might be better but if you are only going to have 500 or so templates then you're fine just using PHP. I made something similar in the past. You just need a web form with inputs for each variable you are using, pass it to backend script which will then use all the variables in the HTML template.

```
<div class="<?Php echo $_POST['mainColor'] == 'green' ? "green-background" : "red-background";
?>">
content <span style='font-size: <?php echo $_POST['fontSize'] ?>px'>goes here</span>
</div>
```

This is a super simple example. You could just set the HTML to a variable instead of doing echos inside of it and then use fwrite() or another function to put the contents into a new html file.

+mK - December 21, 2022, 5:47 pm

(Dec 20, 2022 09:24 PM)Romeos Wrote: [\[CITE\]](#) this is extremely confusing tbh.I do have a way with making shit confusing. :\

(Dec 20, 2022 11:59 PM)Yani Wrote: [\[CITE\]](#) In PHP:

PHP Code

```
<?php
$files= glob('/var/www/css/*.css');
$css_file = $files[random_int(0, count($files) - 1)];

echo "<link type='text/css' rel='stylesheet' href='{ $css_file}' />";

$classes = ['alert-info', 'alert-warning', 'alert-danger', 'alert-success'];
$class = $classes[random_int(0, count($classes) - 1)];

echo "<div class='{ $class}'></div>";
```

Hmm, that's interesting. I didn't think about using an array, but I suppose that could work for specific situations. Thank you!

(Dec 21, 2022 12:24 AM)Shellcode™ Wrote: [\[CITE\]](#) Dynamic content (HTML, CSS, JS) is absolutely doable just by using Raw PHP. Also you will not really find "examples" that do EXACTLY what you have written here anywhere, but instead separate scripts for every moving piece - And you should properly stich/adjust them together.

This is something very doable as I said earlier, but if you are not experienced developer, I strongly suggest you to hire someone.

All I needed to know was if it was possible. Wasn't sure if my search string(s) were shit or not. I'll probably start asking ChatGPT, keep forgetting that bad boy exists.

Thank you!

(Dec 21, 2022 05:59 AM)nodes Wrote: [\[CITE\]](#) Yes, very easy to do with:

- JS
- PHP

You could make it random on page load, or use PHP to generate a template & save it as an HTML file.

For speed, Python might be better but if you are only going to have 500 or so templates then you're fine just using PHP. I made something similar in the past. You just need a web form with inputs for each variable you are using, pass it to backend script which will then use all the variables in the HTML template.

```
<div class="<?Php echo $_POST['mainColor'] == 'green' ? "green-background" : "red-background";
```

?>">

```
content <span style='font-size: <?php echo $_POST['fontSize'] ?>px'>goes here</span>
</div>
```

This is a super simple example. You could just set the HTML to a variable instead of doing echos inside of it and then use fwrite() or another function to put the contents into a new html file.

Thanks for the guidance dude! I'm actually running down this path currently. This is going to take forever, but it'll be worth it.

Hackers - December 21, 2022, 8:41 pm

What I did was made a PHP template manager.

You just make files like 'homepage.txt'

inside of there you do things like this: <div>Hello {%UserName%}, do you like {%thing%}?</div>

Then you just use fopen or file get contents, you have your var like \$templateHtml = (your file).txt then you just do something like this:

PHP Code

```
<?php
$templateHtml = "<div>Hello {%UserName%}, do you like {%thing%}?</div>";
$replaceVars = array(0=> array("var"=>"UserName", "value"=>"mK"), 1=> array("var"=>"thing",
"value"=>"coding"));
foreach($replaceVars as $replacement){
    $templateHTML = str_replace("{%" . $replacement['var'] . "%}", $replacement['value'],
$templateHTML);
}
```

That loops through your variable arrays and auto-replaces all of the values.

You can do it to auto generate CSS too:

PHP Code

```
<?php
$templateHTML = ".userNameClass{
color: {%colorCode%};
}";

$templateHTML = str_replace("{%colorCode%", "#f2f2f2", $templateHTML);
```

I used to just do something like this:

```
run.php
/templates/
```

Inside of templates I'd have index.php.txt, css/ folder, js/ folder, etc.

I would then use something like scandir to get all of the files & folders inside of templates, then auto-replace all my variables using code similar to the stuff above, and then I'd save each file without the .txt extension.

+mK - December 22, 2022, 12:19 pm

I did it! I think. It's better than what the hell I was doing, which was doing a fuck ton of querySelectors in JS and replacing text based on variables at the top of the JS file. Going to be tweaking this to make it less labor some. For example, modify the output HTML file to append the page title variable, so that I can find out which output is for which "theme".

Thank you guys!

Here is the code:

Spoiler ([Click to View](#))Code <!DOCTYPE html>

```
<html lang="en">
```

```
<head>
```

```
<style>
```

```
body {
```

```
  font-family: 'Roboto', sans-serif;
```

```
  background-color: #222; /* Set the background color to a dark shade /
```

```
color: #fff; / Set the text color to white */
```

```
}
```

```
.form-container {
```

```
  max-width: 800px;
```

```
  margin: 0 auto;
```

```
  width: 500px; /* Set the width to 500 pixels */
```

```
  padding: 20px;
```

```
  background-color: #333; /* Set the container's background color to a slightly lighter shade */
```

```
  box-shadow: 0 10px 20px 0 #3498db; /* Add the blue drop shadow */
```

```
}
```

```
header {
```

```
  display: flex;
```

```
  align-items: center;
```

```
  justify-content: space-between;
```

```
  margin-bottom: 20px;
```

```
}
```

```
header h1 {
```

```
  font-size: 2em;
```

```
  font-weight: bold;
```

```
  color: #3498db;
```

```
}
```

```
header nav ul {
```

```
  list-style: none;
```

```
  display: flex;
```

```
}
```

```
header nav li {
```

```
  margin-left: 20px;
```

```
}
```

```
header nav a {
```

```
  font-size: 1.2em;
```

```
  text-decoration: none;
```

```
  color: #3498db;
```

```
}
```

```
.form-container {
```

```
  padding: 20px;
```

```
  border-radius: 10px;
```

```

}

label {
  font-size: 1.2em;
  display: block;
  margin-bottom: 10px;
  color: #ffffff; /* make the text color white */
}

input[type=submit] {
  width: 100%;
  background-color: #3498db; /* make the background color blue */
  color: #ffffff; /* make the text color white */
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

input[type=text] {
  width: 100%;
  padding: 12px;
  border: 1px solid #ddd;
  border-radius: 5px;
  box-sizing: border-box;
  margin-bottom: 20px;
  font-size: 1.2em;
  background-color: #333; /* make the background color dark */
  color: #ffffff; /* make the text color white */
}

.form-container h2 {
  font-size: 1.5em;
  text-align: center;
  margin-bottom: 20px;
  color: #3498db;
}

.form-container form label {
  font-size: 1.2em;
  display: block;
  margin-bottom: 10px;
}

.form-container form input[type="text"] {
  width: 100%;
  padding: 12px;
  border: 1px solid #ddd;
  border-radius: 5px;
  box-sizing: border-box;
  margin-bottom: 20px;
  font-size: 1.2em;

```

```
border: 2px solid #3498db; /* Set the border to a blue color */
}
```

```
label {
    font-size: 1.2em;
    display: block;
    margin-bottom: 10px;
    color: #ffffff; /* make the text color white */
}
```

```
.form-container form input[type="submit"] {
    width: 100%;
    background-color: #3498db;
    color: white;
    padding: 14px 20px;
    border: none;
    border: 2px solid #3498db; /* Set the border to a blue color */
    border-radius: 5px;
    cursor: pointer;
    font-size: 1.2em;
}
```

```
input[type="submit"] {
    color: #fff;
    font-size: 1.2em;
    font-weight: bold;
    padding: 12px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    text-transform: uppercase;
    background: linear-gradient(to right, #3498db, #2980b9);
}
input[type="submit"]:hover {
    background: linear-gradient(to left, #3498db, #2980b9);
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div class="form-container">
```

```
    <h2>HTML Generator</h2>
```

```
<?php
```

```
// Check if the form has been submitted
```

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
```

```
    // Read the contents of the HTML file into a string
```

```
    $html_contents = file_get_contents('index.html');
```

```
    // Loop through the POST data and replace the variables in the HTML contents with their
    corresponding values
```

```
    foreach ($_POST as $var => $value) {
```

```
        $html_contents = str_replace('{%' . $var . '%}', $value, $html_contents);
```

```

}

// Save the modified HTML contents to a new file with the desired title
$new_html_file = 'modified.html';
file_put_contents($new_html_file, $html_contents);
}

// Read the contents of the HTML file into a string
$html_contents = file_get_contents('index.html');

// Find all the variables in the HTML template
preg_match_all('/{%(.*?)%}/', $html_contents, $matches);
$variables = $matches[1];

// Generate the form HTML
$form_html = '<form action="process.php" method="post">';
foreach ($variables as $variable) {
    $form_html .= '<label for="' . $variable . '">{' . $variable . '}%</label><br>';
    $form_html .= '<input type="text" id="' . $variable . '" name="' . $variable . '"><br>';
}
$form_html .= '<input type="submit" value="Generate HTML">';
$form_html .= '</form>';

// Output the form HTML
echo $form_html;

?>
</div>
</body>

</html>

```

Here is the screenshot:

Spoiler [\(Click to View\)](#)

Hackers - December 22, 2022, 10:23 pm

Nice dude! Looks good overall & is exactly what I ended up doing in the past, but you took it a step further to auto generate the entry form. That's super smart. I never thought about auto finding all of the {%-vars%}, so I always manually added fields to my generator forms. I don't have a need for anything like this right now, but I'll definitely remember this.

The only downfall to this method is if your index / css files are pretty much identical except for a few variables. That means you have to change multiple files if you want to change something in the future. If that becomes too burdensome, you could consider having theme settings...

PHP Code

```

<?php
$themes = [
    "light" => [
        "title"=>"My light theme",
        "firstselect"=> "choose it"
    ],
    "dark" => [

```

```
"title"=>"My dark theme",  
"firstselect"=> "choose that"  
]  
];
```

You could then use `$_GET` to print out diff variables in the page based on the theme selection.
`index.php?theme=light`
`index.php?theme=dark`

If this doesn't work for you, you could use `$_COOKIE` to store the current theme for a visitor.

+mK - December 23, 2022, 12:21 am

I think I know what you're saying, but I'm not too sure. You're saying have a toggle to pre-populate the input fields based on whichever theme I choose, right? That would be interesting for slight variations, such as bright default colors, dark default colors, blue base tone, etc.

I wonder if it'd be possible to do an AND operation to select that theme.

I took it a bit further and included pages for JS and CSS manipulation. For example, every single input box on this page is pre-populated with what I think are neutral colors. I also included a color picker that will move based on scroll, so that I don't have to have two separate windows open.

Hackers - December 23, 2022, 5:49 am

(Dec 23, 2022 01:21 AM)+mK Wrote: [\[CITE\]](#) I think I know what you're saying, but I'm not too sure. You're saying have a toggle to pre-populate the input fields based on whichever theme I choose, right? That would be interesting for slight variations, such as bright default colors, dark default colors, blue base tone, etc.

I wonder if it'd be possible to do an AND operation to select that theme.

I took it a bit further and included pages for JS and CSS manipulation. For example, every single input box on this page is pre-populated with what I think are neutral colors. I also included a color picker that will move based on scroll, so that I don't have to have two separate windows open.

Yeah basically what I was saying is that you could modify your script so that it still scans the template files for `{%vars%}` and generates the form. However, instead of the form generating the new `modified.html`, it would save the settings to a thing like `'themeName.json'`. Then when you load up the HTML in future (on `index.php` instead of `index.html`), you could do something where you basically load in `themeName.json` based on a cookie or GET variable, and then you use the variable settings in the JSON (your theme config file) dynamically.

So it's like...

`site.com/index.php?theme=dark`

PHP fetches `themes/dark.json`, and uses `json_decode($filecontents)` to convert to PHP array
Then inside of `index.php` you could just echo out the variable values for each part of the file.

The downside is that if you use a `.css` file then you can't do it this way, but if the CSS is inline (inside of `modified.html` currently) then it would work

I personally always like to avoid duplicate code, so if every modified.html is the exact same structure but with a few different variables in it then it makes sense to convert to an index.php to futureproof the project. However, if you prefer 'scoped' themes that are static (there is a clear performance upgrade there) then keep doing what you are doing because there's nothing really wrong with that method, you just have to manage more files in future if you want to make changes to base template.

Kr?nus - December 23, 2022, 11:00 am

(Dec 23, 2022 06:49 AM)nodes Wrote: [\[CITE\]](#) Yeah basically what I was saying is that you could modify your script so that it still scans the template files for {%vars%} and generates the form. However, instead of the form generating the new modified.html, it would save the settings to a thing like 'themeName.json'. Then when you load up the HTML in future (on index.php instead of index.html), you could do something where you basically load in themeName.json based on a cookie or GET variable, and then you use the variable settings in the JSON (your theme config file) dynamically.

So it's like...

site.com/index.php?theme=dark

PHP fetches themes/dark.json, and uses `json_decode($filecontents)` to convert to PHP array
Then inside of index.php you could just echo out the variable values for each part of the file.

The downside is that if you use a .css file then you can't do it this way, but if the CSS is inline (inside of modified.html currently) then it would work

I personally always like to avoid duplicate code, so if every modified.html is the exact same structure but with a few different variables in it then it makes sense to convert to an index.php to futureproof the project. However, if you prefer 'scoped' themes that are static (there is a clear performance upgrade there) then keep doing what you are doing because there's nothing really wrong with that method, you just have to manage more files in future if you want to make changes to base template.

Be extremely careful if you're to apply this as it's vulnerable to LFI, even if fetching the theme from `$_COOKIE` object. Create an array of possible inputs (light/dark) and always check whether the user input (`$_GET` / `$_COOKIE`) exists in that array.

Hackers - December 23, 2022, 11:54 am

(Dec 23, 2022 12:00 PM)Kr?nus Wrote: [\[CITE\]](#) Be extremely careful if you're to apply this as it's vulnerable to LFI, even if fetching the theme from `$_COOKIE` object. Create an array of possible inputs (light/dark) and always check whether the user input (`$_GET` / `$_COOKIE`) exists in that array.

Yeah, file path traversal is a dangerous beast.

index.php?theme=../.././config.php

Using an array of possible inputs is a good counter to this, but you can also compare file path of your theme folder (/path/to/themes/) to make sure the absolute path of the requested theme file is also a match. There are other ways to protect yourself, but Kronus is right that you would need to do something to prevent LFI.