Surface.

User Guide – Build 0.1.2

1. Introduction

What is Surface?

Surface is framework in PHP to make your works easier with clean codes and security. It is developing fast and a stable version will be available soon.

Who can start using it?

Surface is so designed that anyone with basic OOP knowledge in PHP can start using it to make things awesome!

Which template engine does Surface use?

Surface uses TWIG as its core template engine. You can follow twig docs whenever it is necessary.

Does Surface have caching system?

Well, we are talking about Surface, it provides the best solutions you'll ever find in previous versions of Surface :D. Surface uses advance caching system with TWIG.

How Should I install surface?

For this build, just copy and paste files to your workspace. It should be in root directory.

Please connect to database in app/config/Config.php

Database connection is mandatory in this version.

Hopefully, next versions will have awesome installation procedure and stability to install in any directory of your workspace.

Enough Question and Answers session, Let's get our hands dirty!!

<u>**Basics of our lovely surface!**</u>

What you have to keep in mind:

<u>Structure:</u>

For now I have built Surface so that it should be in root directory of your program. E.g. yoursite.com should contain our main application.

'app' folder contains all our application data.

'system' folder contains all system data. You are strongly recommended not to mess things up with that folder. You have nothing to do with that if you are not contributing in development of Surface.

'assets' folder contains all our css, js, image and other files.

Okay so what are we concerned about, when we are developing app using Surface?

Keep in mind:

We have to code in 'app' folder and add our customization files in assets folder (Recommended but you can use any other folder as well :D).

Starting:

app/routes/Routes.php contains all the allowed routes. This is basically starting point.

$this->routes stores all our routes.

Example:

To create yoursite.com/example

Step 1:

Create app/Controllers/ExampleController.php

Put following code:

```
<?php

namespace app\controllers;


use \system\controllers\Controller;

//Every controller should have above code

/* Class name should be same as file name. eg. ExampleController.php should
have ExampleController class which should extend
\system\controllers\Controller */




/**

* @Controller  ExampleController

*/

class ExampleController extends Controller

{

	public function exampleMethod()

	{

		echo 'You are in exampleMethod! Great!';

}




} //End of ExampleController
```

Step 2:

In app/routes/Routes.php

Check for $this->routes and do something like this:

```
$this->routes = [

        '/' => ['uses' => 'Welcome@welcomeToSurface'],

        'example' => ['uses' => 'ExampleController@exampleMethod'],

];
```

Step 3:

Seems like you're in hurry!

You've already done B-)

Just check yoursite.com/example now!

Easy, isn't it?

With Surface, everything is going to be easy! Just take some time to chill!

More useful stuff:

app/controllers this folder contains our controllers.

app/models this folder contains our models.

app/views this folder contains our view files.

Last time we gave output from exampleMethod directly with echo. This is no good choice. Instead, we will use our view function.

In exampleMethod

Change the code to following:

```
$data['somevar'] = 'This is value of somevar.';

return view('firstview.php', $data);
```

<u>Note that second parameter of view is optional.</u>

Create app/views/firstview.php (you can also create .html, .tpl file)

Add following code:

```
This is some html text. We can use html here. This is our firstview. Value of somevar- {{somevar}}
```

Now check yoursite.com/example

For further enquiries to use codes in view docs, please use TWIG Docs.

Okay, for this user guide, I will wrap things little bit quicker.

Some features available in Surface:

- *MVC Pattern*
- *Twig Templating (Caching, Layouts and lots of other stuff!)*
- *Middlewares (Yes!, you heard that)*
- *Advanced Routing system (More like REST, will be more powerful in further versions)*
- *Libraries (inbuilt and external, I have added Validation library, docs will be available shortly!)*
- *More…*

More in next chapter of user guide. Contact me for suggestions, errors, bugs.