Tesseract LASAGNA: MVP PWA Framework

version: 2.0 beta (2022-03-10)

Concept

Tesseract LASAGNA is a fast, modern and modular PHP OOP framework for rapid prototyping of **Progressive Web Apps** (PWA). Tesseract uses *Google Sheets CSV export* as a data input and builds the **Model** from CSV layers (hence the LASAGNA codename).

There are **Presenters** used to process the **Model** and to export the result in TEXT, JSON, XML or HTML5 format (or any other custom format). **View** is built as a set of *Mustache templates* and *partials*.

Tesseract is based on **Composer components**, complex **RESTful API**, incorporates a **command line interface** and **Continuous Integration** testing.

Tesseract uses no database structures, so it is quite easy to implement all types of scaling and integrations. Acccess is based on **OAuth 2.0** and the Halite **encrypted master key**.

Installation

PHP Source

Clone the repository https://github.com/GSCloud/lasagna

```
git clone https://github.com/GSCloud/lasagna.git
```

and run:

```
cd lasagna; make install
```

Docker Container

Run:

```
docker run --rm -d --name lasagna -p 9000:80 gscloudcz/tesseract-lasagna:latest
```

Run updater:

```
docker exec lasagna make du
```

and visit:

```
http://localhost:9000/
```

Update

Go to the Lasagna directory and run:

```
make update
```

Basic Functionality

Index

Tesseract starts parsing the **www/index.php** file, that's targeted at the Apache level via **.htaccess** configuration file using *mod_rewrite*. **Index** can contain various constant overrides. **Index** then loads the **Boostrap.php** core file from the root folder.

Bootstrap

Bootstrap sets the constants and the application environment, **Nette Debugger** is also instantiated on the fly. Bootstrap loads the **App.php** core file from the app folder.

App

App processes the application configuration files (public and private), sets caching mechanisms (optional Redis database support), configures URL routing, emmits CSP headers and sets the **Model** (multi-dimensional array).

App loads the corresponding *presenter* based on the actual URI route. It can also run a *CLI presenter*, if the CLI is detected. When the *presenter* returns an updated Model, the output is echoed and final headers are set (including some optional debugging information). Runtime ends here.

Router

Router is a part of the **App** script and is defined by joining (*array replace recursive*) several routing tables (in NE-ON format) in the /**app** folder.

- **router_defaults.neon** - default values (global)
- **router_core.neon** - core Tesseract funcionality (global)
- **router_admin.neon** - administrative routes (global)
- **router_extras.neon** - extra features (optional)
- **router_api.neon** - API calls go here
- **router.neon** - all the web app pages

Presenter

Presenter is a subclass instance based on an *abstract class* **APresenter.php** and defines at least the *process()* method, that is called from the **App**. The *process()* method can either output the resulting data or return it encapsulated inside the Model back to the **App** for rendering.

The instance is created and data processed like this:

```
$app = $controller::getInstance()->setData($data)->process();
```

Getting the output for enduser:

```
echo $app->getData()["output"] ?? "";
```

How to display custom presenter in Lynx terminal using a CLI helper (default = home):

```
./cli.sh app '$app->show();' | lynx -dump -stdin
```

```
./cli.sh app '$app->show("home");' | lynx -dump -stdin
```

How to display core module output using a CLI helper (default = PingBack):

```
./cli.sh app '$app->core();'
```

```
./cli.sh app '$app->core("PingBack");'
```

```
./cli.sh app '$app->core("GetTXTSitemap", ["base"=>"https://google.com/"]);'
```

```
./cli.sh app '$app->core("GetXMLSitemap", ["base"=>"https://google.com/"]);'
```

API

API is generated from the routing tables on the fly.
See the live demo at this URL: https://lasagna.gscloud.cz/api

Command Line Interface

Makefile

Run `make` to see the inline documentation.

Bootstrap CLI

```
./cli.sh <command> [<parameter> ...]
```
or
```
php -f Bootstrap.php <command> [<parameter> ...]
```

```
app '<code>'    - run inline code
clear           - alias for clearall
clearall        - clear all temporary files
clearcache      - clear cache
clearci         - clear CI logs
clearlogs       - clear logs
cleartemp       - clear temporary files
doctor          - check system requirements
local           - local CI test
prod            - production CI test
unit            - run Unit test (TBD)
```

Examples:

```
./cli.sh clear
```

```
./cli.sh app
```

Filesystem Hierarchy

- **apache/** - Apache configuration example
- **app/** - Presenters and NE-ON configurations
- **bin/** - bash scripts for Makefile
- **ci/** - Continous Integration logs
- **data/** - private data, encryption keys, CSV imports, etc.
- **doc/** - phpDocumentor generated documentation
- **docker/** - files to be inserted into the Docker container
- **logs/** - system logs
- **node_modules/** - Node.js modules used by Gulp
- **temp/** - temporary files, Mustache compiled templates
- **vendor/** - Composer generated vendor classes
- **www/** - static assets
  - **www/cdn-assets/** - repository version hash-links to www/
  - **www/css/** - CSS classes
  - **www/docs/** - link to doc/
  - **www/download/** - downloadable files
  - **www/epub/** - ePub files
  - **www/img/** - images
  - **www/js/** - JavaScript files
  - **www/partials/** - Mustache partials
  - **www/summernote/** - Summernote editor
  - **www/templates/** - Mustache templates
  - **www/upload/** - uploads via administration panel
  - **www/webfonts** - fonts

Model

Tesseract Model is a multi-dimensional array. You can list the model keys easily like this:

```
./cli.sh app 'dump(array_keys($app->getData()));' | more
```

or dump the whole model: `./cli.sh app 'dump($app->getData());' | more`

Model is supported by two methods: `getData()` and `setData()`. Both methods accept the *dot notation*, e.g.:

```
./cli.sh app 'echo $app->getData("router.home.view");'
```
home

```
./cli.sh app 'echo $app->getData("cfg.project")'
```
LASAGNA

Constants

Tesseract specific constants can be listed by a command:

```
./cli.sh app '$app->showConst()'
```

Constants can be overriden in **www/index.php**, otherwise they are defined in the Boostrap and the App.

Bootstrap.php

- **APP** - *application* folder
- **AUTO_DETECT_LINE_ENDINGS** - Tesseract detects line endings by default
- **CACHE** - *cache* folder
- **CLI** - TRUE if running in terminal mode
- **CONFIG** - *public configuration* file
- **CONFIG_PRIVATE** - *private configuration* file
- **CSP** - *CSP HEADERS* configuration file
- **DATA** - *application data* folder, also *private data* goes here
- **DEBUG** - TRUE if debugging is enabled
- **DEFAULT_SOCKET_TIMEOUT** - 30 seconds timeout
- **DISPLAY_ERRORS** - Tesseract displays errors by default
- **DOWNLOAD** - *download* folder
- **DS** - operating system *directory separator*
- **ENABLE_CSV_CACHE** - enable use of extra *curl_multi CSV cache*
- **LOCALHOST** - TRUE if running on a *local server*
- **LOGS** - *log files* folder
- **PARTIALS** - *Mustache partials* folder
- **ROOT** - *root* folder
- **TEMP** - *temporary files* folder
- **TEMPLATES** - *templates* folder
- **TESSERACT_END** - execution UNIX time end
- **TESSERACT_START** - execution UNIX time start
- **UPLOAD** - *upload* folder

- **WWW** - *static assets* folder, also the *Apache root*

App.php

- **CACHEPREFIX** - cache name prefix
- **DOMAIN** - domain name
- **SERVER** - server name
- **PROJECT** - project name (higher level)
- **APPNAME** - application name (lower level)
- **MONOLOG** - Monolog log filename
- **GCP_PROJECTID** - Google Cloud Platform (GCP) project ID
- **GCP_KEYS** - GCP auth keys JSON base filename (in **app**/)

Administration

Authentication

Tesseract login is based solely on the **Google OAuth 2.0** client right now.

When the user logs in, a master key - Halite encrypted cookie is created and set via HTTPS protocol (strict). This cookie is protected from tampering and its parameters can be modified in the administration panel, or remotely via authenticated API calls.

The logging is available if OAuth parameters are set!

There is no database of connections or authenticated users at all. The default login URL is **/login** and the default logout URL is **/logout**.

> *Halite* is a high-level cryptography interface that relies on libsodium for all of its underlying cryptography operations. Halite was created by Paragon Initiative Enterprises as a result of our continued efforts to improve the ecosystem and make cryptography in PHP safer and easier to implement.

To display the structure of the unencrypted master key, run the following command:

```
./cli.sh app 'dump($app->getIdentity())'
```

More detailed information can be obtained this way:

```
./cli.sh app 'dump($app->getCurrentUser())'
```

Note: These commands always return the string ``XX'' for the country code, because this information is obtained from the Cloudflare header itself.

Permissions

Tesseract has built-in three basic permission levels, that can be easily extended.

Core levels are:

1. **admin** - superuser,
2. **editor** - can refresh data and edit articles,
3. **tester** - no elevated permissions,
4. **authenticated user** - rights the same as level 3, and
5. **unauthenticated user** - unknown identity.

Remote Calls

Remote calls are handled by the *AdminPresenter*, administrator can generate the corresponding URIs in the administration panel.

- **CoreUpdateRemote** - download CSV files and rebuild the data cache
- **FlushCacheRemote** - flush all caches
- **RebuildAdminKeyRemote** - recreate random admin key (authentication of remote calls)
- **RebuildNonceRemote** - recreate random nonce (identity nonce)
- **RebuildSecureKeyRemote** - recreate random secure key (cookie encryption)

Automation on localhost is possible by using the **admin key** as a `?key=` parameter in a curl call.
The key is readable for root or www-data group:

```
cat data/admin.key
```

Core Features

Versioning

All static assets are automatically versioned by using a git version hash. This hash is used to generate a symbolic link in the **www/cdn-assets** folder.

The symbolic link looks like this:

```
./cli.sh app 'echo $app->getData("cdn")'
```
/cdn-assets/4790592b350262b8e1960a96a097de0af1828532

and can be used to version the assets in Mustache template like this:
```
<image src="{{cdn}}/img/logo.png">
```

Web Pages

TBD

Translations

TBD

PWA Manifest

TBD

Service Worker

TBD

Icons

TBD

Fonts

TBD

Sitemaps

Tesseract generates TXT and XML sitemaps based on the routing tables.

https://lasagna.gscloud.cz/sitemap.txt

https://lasagna.gscloud.cz/sitemap.xml

CSP Headers

You can define headers for *Content Security Policy* in **app/csp.neon** file.

Extra Features

Articles

TBD

QR Images

The route goes as **qr/[s|m|l|x:size]/[:trailing]**.
Hello World example: https://lasagna.gscloud.cz/qr/s/Hello%20World

EPUB Reader

TBD

Pingback Monitoring

Pingback service posts some detailed information about the state of the server.
See the live demo at this URL: https://lasagna.gscloud.cz/pingback

Data Exports

Article data can be exported based on the article language (CS), profile (default) and page ID (use `home' for the homepage).

https://lasagna.gscloud.cz/cs/exportHTML/default/home

https://lasagna.gscloud.cz/cs/exportHTML/default/id/demo

Android App Extras

TBD

What's next?

CURRENT: Known Bugs

- **adbario/php-dot-notation** package contains PHP 8.1 deprecation bugs that can be fixed by overwriting the **vendor/adbario/php-dot-notation/src/Dot.php** file with **app/Dot.php** temporary fix

FUTURE: TODO Implementations

- **Multi-site** - multiple sites support (partially ready)
- **Dark Mode** - set UI in the dark
- **Tesseract Configurator** - web based configuration UI
- **Links Manager** - Links manager UI
- **SEO Manager** - SEO manager UI
- **Administration UI** - administration UI reborn