Tesseract LASAGNA: MVP PWA Framework

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 exports* as a data input, it builds the Model from CSV layers (hence the LASAGNA codename).
Abstract based **Presenters** are used to process the **Model** and to export resulting data in TEXT, JSON, XML or HTML5 formats (or any other custom format). **View** is built as a set of Mustache templates and *partials* (Mustache can be also rendered in the browser via JavaScript).
Tesseract is *Composer components* based, the Model defines a complex **RESTful API**, has a *command line interface* (CLI) and incorporates *continuous integration* (CI) testing.
Tesseract uses no classical database models and structures, so it is quite easy to implement all types of scaling and integration. The acccess model is based on the **master key encrypted cookie**.

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 definitions and overrides. **Index** then loads the **Boostrap.php** core file from the aplication root folder.

Bootstrap

Bootstrap sets core constants and the application environment, **Nette Debugger** is also instantiated on the fly. Bootstrap then loads the **App.php** core file from the **app/** folder (the location can be overriden via a constant).

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 core **Model** (multidimensional array). **App** then loads the corresponding *presenter* based on the actual URI and the coresponding 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.

Presenters

Presenters are subclass instances based on an *abstract class* **APresenter.php** and define 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.

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

Constants

Bootstrap.php

- **APP** - application folder
- **CACHE** - cache folder
- **CLI** - TRUE if running from command line interface
- **CONFIG** - configuration file
- **CONFIG_PRIVATE** - private configuration file

- **CSP** - CSP HEADERS configuration file
- **DATA** - application data folder
- **DOWNLOAD** - download folder
- **DS** - directory separator
- **ENABLE_CSV_CACHE** - enable use of extra curl_multi cache for CSV
- **LOCALHOST** - TRUE if running server locally
- **LOGS** - log files folder
- **PARTIALS** - templates partials folder
- **ROOT** - root folder
- **TEMP** - temporary files folder
- **TEMPLATES** - templates folder
- **UPLOAD** - upload folder
- **WWW** - assets folder

App.php

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

Administration

Login and Logout

Tesseract login is based solely on the **Google OAuth 2.0** client right now.
When the user logs in, a special encrypted cookie - a master key - is created and set via HTTPS protocol. This cookie is protected from tampering and its parameters can be modified in the administration panel, or remotely via authenticated API calls.
There is no database of connections or authenticated users at all. The default login URL is **/login** and the default logout URL is **/logout**.

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.

Basic Features

Sitemaps

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

CSP Headers

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

Extra Features

QR Image

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

EPUB Ebook Reader

TBD

WYSIWYG Articles

TBD

Pingback Monitoring

See the live demo at this URL: [https://lasagna.gscloud.cz/pingback]