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 (can be also rendered in the browser).
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.

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 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'' the updated Model back, 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**

Filesystem Hierarchy

- **apache/** - Apache configuration example
- **app/** - Presenters and NE-ON configurations
- **bin/** - bash scripts for Makefile
- **ci/** - Continous Integration output
- **data/** - private data, encryption keys, CSV imports…
- **doc/** - phpDocumentor generated documentation
- **docker/** - files to be inserted into the Docker container
- **logs/** - logs
- **node_modules/** - Node.js modules for Gulp
- **temp/** - temporary files, Mustache compiled templates
- **vendor/** - Composer generated vendor classes
- **www/** - static assets (CSS, JS, fonts, images) and CDN hash-links

Administration

Login

Tesseract login is based solely on the **Google OAuth 2.0** client right now. When the user logs in, a special encrypted cookie working as a master key is created and set via HTTPS protocol. There's no database of connections or users at all.

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.