Technical Manual

Project Zombie

LE TEMPS

Proposed by :

M. Romain BEAUGRAND
Mail : rbeaugrand@seriel.net

M. Mahdi LARIBI
Mail : mlaribi@seriel.net

Summary

1 INTRODUCTION

This document is a technical manual on how to administrate and update Zombie. This manual is a document delivered by Seriel for the Zombie project. It describes which technologies have been used, the alterable parameters and the automatic scripts for software. A specific part of the manual explains the procedure of adding modules in the Zombie app by a PHP Symfony developer.

2 TECHNOLOGIES

1- *Main Technologies*

The main technologies used for Zombie are the most used in web application developement. They give an excellent documentation. Zombie is developed in PHP 5.6 and the Symfony framework (most used Framework). The link between the server and the database is managed by Doctrine and pdo_mysql. The database is ran by MySQL with UTF8 format and MyISAM type. The front-end is developed in HTML5/ CSS3/ Javascript (Jquery).

2- *External API*

Zombie uses many externals APIs tor get external data or to use a development functionality.

- FosUserBundle is a Symfony bundle that serves to quickly develop a secure application with login / password.
- AppliToolBox is a Symfony bundle edited by Seriel. It is a toolbox for application development.
- ChartBeat API is a web service that serves to have data analytics of a website.
- GoogleAnalytics API is a web service that serves to have data analytics of a website.
- DonReach Api is a web service that serves to have data analytics on social networks.
- Dandelion Api is a web service of semantic analysis.
- Snowball stemming is a PHP extension for stemmer (semantic search).

Other external APIs can be added in Zombie (view part Add module).

3 SETTINGS

Zombie has been developed with the goal of being easily configurable. The parameters are in the application, in the file “parameters.yml” or in the database.

1- *App Settings*

The settings are available for the user when logged in with the administrator profile. These parameters are located in the administrator menu.

- The user tab serves to add or update a Zombie user. The administrator has a choice to enable or to disable an user. He can update the user profile.
- The profile tab serves to modify or add a profile with user permissions. The permissions are divided into two categories, the first one is a permission for the app navigation. The second one « Type de données » is a permission for the database's data. The profile tab serves also to set the main page.

2- *Settings in file “parameters.yml”*

This setting is linked with the Symfony technology. The file must be filled with parameters in the Yml format (Caution, tab character is prohibited). If file's format is wrong, then Zombie will not be operational. The file's path in : « app/config/parameters.yml ». The comments in file explain the parameters.

As each update is made, the administrator will use Symfony scripts (see commands section) to generate the cache of app.

3- *Database settings*

The project Zombie needs data to be functional. Here are the tables linked with the Zombie configuration (Caution, a save is necessary with each update made) :

- Credential : List of permission types used in PHP code.
- Entite : Description of organization, here « Le temps » (only 1 row)
- Societe: Description of organization, here « Le temps » (only 1 row)
- seriel_user : User's login and password information (Modify in Zombie interface)
- Individu : Others Zombie user data. Individu and Seriel_user are linked. (Modify in Zombie interface)
- individu_entite : User linking with their respective entities (only there « Le temps ») (Modify in Zombie interface)

- individu_profil : User linking with their respective profile. (Modify in Zombie interface)
- Profil: List of profile (Modify in Zombie interface)
- profil_credential : Profile permission type definition(Modify in Zombie interface)
- interface_param : Main page parameters for each profile with search table and reporting table (« recherche_sauvegarde » and « reporting_sauvegarde »)

4 COMMANDS

Many PHP commands are available in Zombie, these commands come form the Symfony technology or Zombie. A daily script is available for inporting all external Zombie data (path of script : « app/scripts/importDay.sh »). All commands must be run in Zombie repertory.

1- *Symfony commands*

List of Symfony command for a good use of Zombie. (See Symfony's official documentation) :

- php bin/console cache:clear : Updates cache for the Zombie commands
- php bin/console cache:clear –-env=prod : Updates cache for the website Zombie
- php bin/console assets:install –-env=prod : Re-generates css, javascript and picture files (First step)
- php bin/console assetic:dump –-env=prod : Re-generates css, javascript and picture files (second step)
- php bin/console doctrine:schema:update --force : Updates the Database structure.

2- *Zombie commands*

List of Zombie commands with a small description :

- php bin/console zombie:cmd cleanTmpLinks : Cleans the temporary table article_tmp_link
- php bin/console zombie:cmd cleanSearchHelper : Cleans the temporary table `search_helper`

Chartbeat:

- php bin/console zombie:chartbeat getReportForDay 2017-01-01 : Imports raw data of chartbeat API for day 2017-01-01.
- php bin/console zombie:chartbeat matchArticlesWithChartbeatReports : Links the data of chartbeat with Article.
- php bin/console zombie:chartbeat pre_calculate : First step for pre-calculating measures and indicators of chartbeat
- php bin/console zombie:chartbeat calculate : Second step for calculating measures and indicators of chartbeat (First step is obligatory).

CrossIndicator:

- php bin/console zombie:crossindicator AllcalculateIndicators : Calculates all cross indicators for all articles.

Dandelion:

- php bin/console zombie:Dandelion extractEntitiesFromArticle 2017-01-01 : Extracts the Dandelion API data for the articles published at a given date.
- php bin/console zombie:Dandelion calculate 2017-01-01 2017-01-02 : Processes the API Dandelion data.

DonReach:

- php bin/console zombie:DonReach getMeasuresForDay 2017-01-01 : Extracts the DonReach API data for the day 2017-01-01.
- php bin/console zombie:DonReach calculate : Calculates all indicators and measures of DonReach for all articles.

Googleanalytics:

- php bin/console zombie:googleanalytics calculateMeasure 2017-01-01 2017-01-02 : Extracts the Google Analytics API data for the day 2017-01-01.
- php bin/console zombie:googleanalytics calculateMetrics : Calculates all indicators and measures of Google Analytics for all articles.

Relatedword:

- php bin/console zombie:relatedword import 2017-01-01 2017-01-01 : Generates the data of articles published between 2017-01-01 and 2017-01-01 for semantics search.
- php bin/console zombie:relatedword getrelatedword word1 20 : Returns 20 words related to « word1 ».
- php bin/console zombie:relatedword getrelatedword word1 word2 40 : Returns the similarities between 2 words (40 is a depth of the semantic search).

Trend:

- php bin/console zombie:trend importTrendsGoogle 2017-01-01 2017-02-02 : Saves the GoogleTrends for the days between 2017-01-01 2017-02-02.
- php bin/console zombie:trend getTrendsFutur 2030-01-01 100 : Returns the future Trends (100 is a precision)

5 ADDING MODULES

The Zombie project has a structure that allows it to add modules. To add modules, you can add Symfony bundle with the following structure type :

1- Metrics module

- Create a Symfony Bundle in the project.
- Create an Entity with the parent class Seriel\AppliToolboxBundle\Entity\RootObject which implements the interface ZombieBundle\API\Entity\ArticleMetrics. This class must have the desired measures and the desired indicators.
- Define the following functions getAllMeasures(), getAllIndicators(), getMeasure($measure), getIndicator($indicator), getAllIdIndicators(), getAllLogoIndicators(). (See the example in entity Seriel\GoogleAnalyticsBundle\Entity\GoogleAnalyticsArticleMetrics)
- Add the annotations @SER\ListeProperty (Table view) and @SER\ReportingDataProperty (Statistics) at getters functions on desired values. (See the example in entity Seriel\GoogleAnalyticsBundle\Entity\GoogleAnalyticsArticleMetrics)
    - First variable : Id
    - Labellogo : Define the logo path
    - Label : Define label
    - class_supp : If your value is an indicator then fill the value with « indicator » otherwise don't use the parameter.
    - credential : Can be used to restrain access to the value.
- Create a Symfony service that implements the interfaces ZombieBundle\API\Managers\ManagerMetrics (Link with Articles) and ZombieBundle\API\Managers\ManagerStateData (Report of import data).
- Define the function getMetricsObjectForArticle so it returns your entity with one article object in parameters. Define the function getStateImports() for it returns a list of state (class ZombieBundle\API\Entity\StateImport).
- Create a file parameters.yml in your bundle. Fill in the file with the following code :

```
parameters:
  zombie.modules.{{ name_module }}:
    service: {{ name_service }}
```

metrics_object_class: {{ class_entite }}

StateMetrics: {{ name_service }}

- Updating the file « app/config/config.yml » :
    - Include your file parameters.yml and services.yml of your bundle
    - Add CSS and Javascript files in the Assetic configuration.
- Add the name of your service in file parameters of CrossIndicator bundle on the parameter « references_manager_metrics ». This adds the indicators in the CrossIndicator bundle.

2- *Semantics module*

- Create a Symfony Bundle in the project.
- Create a Symfony service that implements the folllowing interfaces ZombieBundle\API\Managers\ManagerSemantique and ZombieBundle\API\Managers\ManagerStateData (Report of import data).
- Define the functions getArticlebySearchSemantiqueWithSubjects($subject) and getArticlebySearchSemantique($text) so they return a table with the following format : (id => valueId, totalweight=> valueTotalweight). id => Id of article, totalweight => the weight of article in the semantic search.
- Define the function getSimilarity($text1,$text2,$depth), the function must return a percentage of similarity between 2 string.
- Define the function getStateImports() so it returns a list of state (use class ZombieBundle\API\Entity\StateImport).
- Create a file parameters.yml in your bundle. Fill in the file with the following code :

parameters:

zombie.modules.{{ name_module }}:

service: {{ name_service }}

search_semantique*: {{ name_service }}*

StateSemantique: {{ name_service }}

- Updating the file « app/config/config.yml » :
    - Include parameters.yml and services.yml of your bundle
    - Add CSS and Javascript files in the Assetic configuration.

3- *Trends module*

- Create a Symfony Bundle in the project.
- Create a Symfony service that implements the following interfaces ZombieBundle\API\Managers\TrendsModule and ZombieBundle\API\Managers\ManagerStateData (Report of import data).
- Define the functions getTrends($qte, $startDate, $endDatel, $options) and getTrendsFutur($date,$precision,$options) so they return a table with following format : (trend => weight). Trend => Text of trend, weight => the weight of trend in the day.
- Define the function getStateImports() so it returns a list of state (use the following class ZombieBundle\API\Entity\StateImport).
- Create a file parameters.yml in your bundle and in path « Resources/config/parameters.yml » Fill the file with the following code :

```
parameters:
  zombie.modules.{{ name_module }}:
    service: {{ name_service }}
    service_trend: {{ name_service }}
    semantique_similarity: {{ name_service_semantic }}
    StateTrends: {{ name_service }}
```

- Updating the file « app/config/config.yml » :
  - Includes parameters.yml and services.yml of your bundle
  - Add CSS and Javascript files in the Assetic configuration.

4- *CrossTrends module*

- Create a Symfony Bundle in the project.
- Create a Symfony service that implements the interface ZombieBundle\API\Managers\TrendsModule.
- Define the functions getTrends($qte, $startDate, $endDate, $options) and getTrendsFutur($date,$precision,$options) for they return a table with following format : (trend => weight). Trend => Text of trend, weight => the weight of trend in the day.
- Create a file parameters.yml in your bundle and in path « Resources/config/parameters.yml » Fill the file with the following code :

```
parameters:
  zombie.modules.{{ name_module }}:
    service: {{ name_service }}
    service_trend: {{ name_service }}
    semantique_similarity: {{ name_service_semantic }}
    percent_agregation : {{ percent_min_of_similarity }}
```

- Updating the file « app/config/config.yml » :
  - Includes parameters.yml and services.yml of your bundle
  - Add CSS and Javascript files in the Assetic configuration.

5- CrossIndicator module

- Create a Symfony Bundle in the project.
- Create an Entity with the parent class Seriel\AppliToolboxBundle\Entity\RootObject which implements the interface "ZombieBundle\API\Entity\ArticleMetrics". This class must have the desired measures and desired indicators.
- Define the following functions getAllIndicators(), getIndicator($indicator), getAllIdIndicators(), getAllLogoIndicators(). (See the example in entity Seriel\CrossIndicatorBundle\Entity\CrossIndicatorArticle)
- Add the annotations @SER\ListeProperty (Table view) and @SER\ReportingDataProperty (Statistics) at the desired values in the get functions. (See the example in entity Seriel\CrossIndicatorBundle\Entity\CrossIndicatorArticle)
  - First variable : Id
  - Labellogo : Define the logo path
  - Label : Define label
  - class_supp : If your value is an indicator then fill in the value with « indicator » otherwise don't use the parameter.
  - credential : Can be used to restrain access to the value.
- Create a Symfony service that implements the following interfaces ZombieBundle\API\Managers\ManagerCrossMetrics (this gets all the indicators for all the modules), ZombieBundle\API\Managers\ManagerMetrics (Link with the articles) and ZombieBundle\API\Managers\ManagerStateData (Report of import data).

- Define the function getMetricsObjectForArticle so it returns your entity with one article object in parameters. Define the function getStateImports() so it returns a list of state (class ZombieBundle\API\Entity\StateImport).
- Define the functions getIndicator($article,$indicator), getMeasure($article,$measure), getAllIndicators($article), getAllMeasures($article) (See the example in service Seriel\CrossIndicatorBundle\Managers\CrossIndicatorManager).
- Create a file parameters.yml in your bundle. Fill in the file with the following code :

  ```
  parameters:
    zombie.modules.{{ name_module }}:
      service: {{ name_service }}
      metrics_object_class: {{ class_entite }}
      metrics_cross_object_class : {{ class_entite }}
      StateMetrics: {{ name_service }}
  ```
- Updating the file « app/config/config.yml » :
  - Includes parameters.yml and services.yml of your bundle
  - Add CSS and Javascript files in the Assetic configuration.