**The Thought-Yards Framework**

**The Design and Approach Document**

1. Introduction
2. Pre requisites
3. Installation
4. Components
5. Application and Command Usage
6. Framework and others.
7. Assumptions
8. Thanking & References
9. **Introduction**

To achieve the task, the design thought was incepted to create the CLI Application which can cater the “Mikko Test” for salary disbursement. To start with the application has both CLI as well as the Web Version. The program from conception, inception and implementation is written, designed and architected to cater the business needs of any organization using MVC patterns, dependencies injections using service containers, Yaml Pattern files, Auto loading using composers, browser terminal.

I preferred to create a new Framework in the Salary Disbursement application for “Mikko Test” rather than using already available in the market. This framework supports MVC for CLI and Web, both.This framework is highly sustainable over adding new libraries and using them by simple configuration in Yaml files and container as a services.

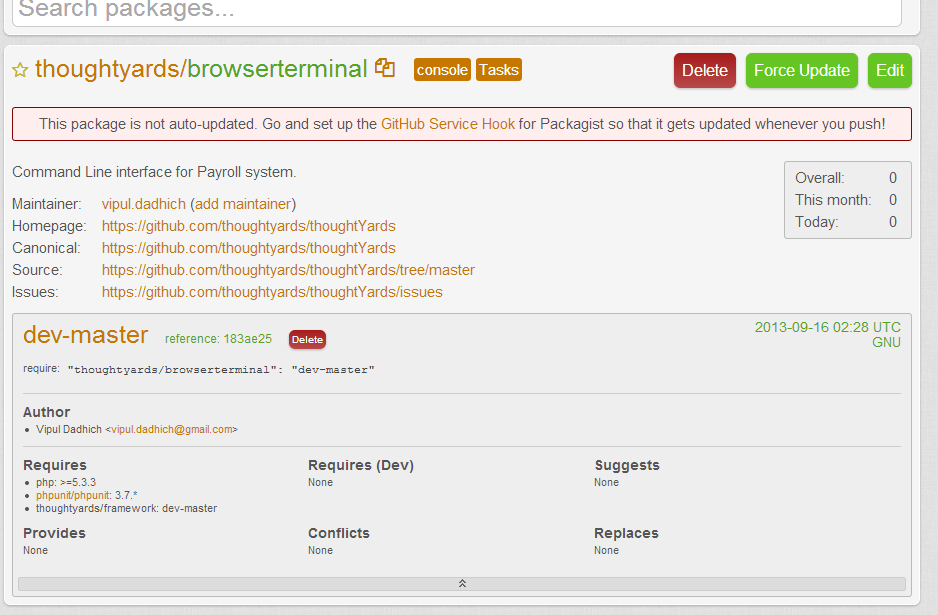
The framework has been named affectionately as ThoughtYards.This framework currently is in very nascent stage being it just 7 days old hence it is not tested on all the aspects and it requires to brainstorm to make it production ready.However it works fine on Happy Path used for “Mikko Test”.

1. **Pre requisite**
2. Composers – <http://composer.org>
3. JIT Hub
4. PHP 5.3 and greater
5. Access to terminal window
6. **Installation**

* Download composer.phar in ur route directory
* Copy composer .jason in the same level
  1. Way 1- Code has been pushed to gitHub. You can access the code @

<https://github.com/thoughtyards/thoughtYards>

* 1. Way -2The code can be auto deployed through <http://packagist.org/> as well



<https://packagist.org/packages/thoughtyards/browserterminal>

include the package in you root folder. Add the dependency by adding

require "thoughtyards/browserterminal": "dev-master" line in your composer.json

* Run the command composer install
* Require dependencies thoughtyard/framework
  1. php 5.3/
  2. php unit/php unit
* Create the virtual host file and copy paste the text in the file

<VirtualHost \*:80>

ServerAdmin postmaster@mindtree.com

DocumentRoot "C:/xampp/htdocs/root/ TheMainPath /"

ServerName thougtyards-mikko. demo

<Directory "C:/xampp/htdocs/root/TheMainPath/">

Options Indexes FollowSymLinks Includes ExecCGI

AllowOverride All

Order allow,deny

Allow from all

</Directory>

</VirtualHost>

* Activate the site

/etc/hosts entry

127.0.0.1 thougtyards-mikko.demo

* Browser Exception to proxy
* Ready to go

1. **Components**

* CSV parser
* Browser based terminal akka Shell Commander
* Yaml file parcel
* Dependency injection using AOP
* Consol command kit and Webkit (Integrated with framework)
* Logger

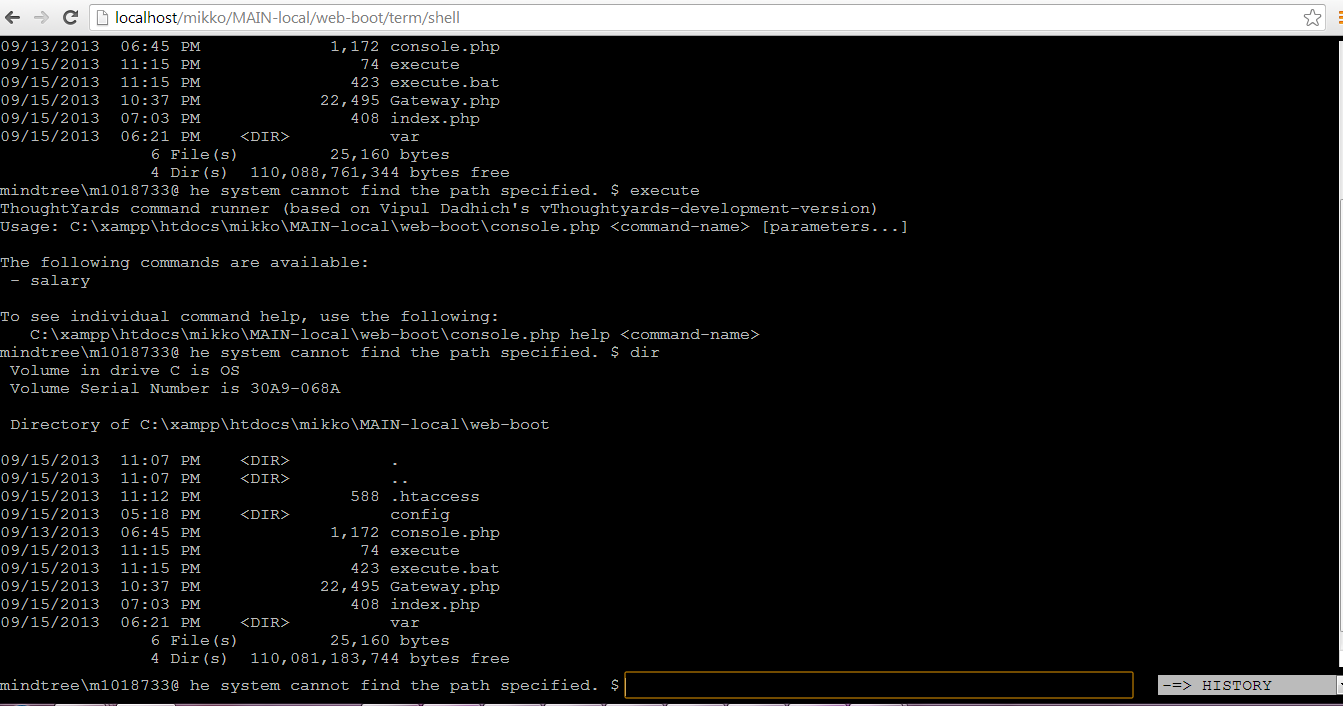
1. Application and Command Usage

The application is named as Terminal Kit/ payroll and the entire application source code lies inside src/terminalkit/payroll.

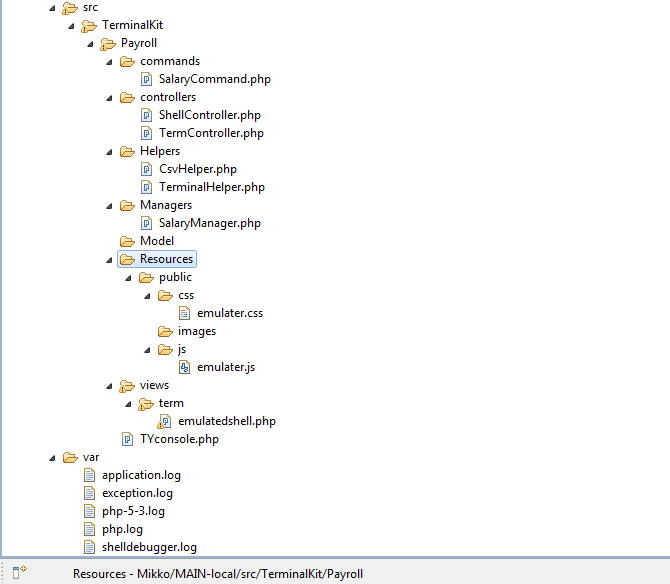
The application supports browser based terminal which exactly has same behavior of terminal. The terminal can be accessed through the below mentioned URL :

<http://thoughtyards-mikko.demo/>

This will render the shell in the browser using MVC patterns.

This shell emulator will be used to access all the commands available in terminal kit or running any OS level operating system command. This tool is very beneficial in cases where the user have very limited access to the Web servers and along with ThoughtYard and model view Command it provides full control over the application.

Folder structure of the module terminal kit payroll: This does not include the framework structure.



Explanation of Directory structure:

1. Controllers- It is the collection of controllers and actions, rerouted from ThouhtYards kinetic/controller components and other base classes. Behind the scenes framework is taking care of passing headers and converting the request into objects.The generic methods automatically renders the presentation layer through generic methods.

Note:Framework will resolve Layout XML files to maintain the dependencies like assets and block related assets.

1. Commands- It is the collection of commands and actions, rerouted from ThouhtYards kinetic/commands components and other base classes. Behind the scenes framework is taking care of passing arguments and converting the request into objects.The generic methods automatically renders the output in command line interface through generic methods.
2. Helper- Helper consists of helper classes which caters special utilities like CSV parsing and terminal command parsing to generate output in presentation layer.

Terminal Helper-This works on Ajax based responses which works on session handlers and displays the result output of command written in shell commander (Browser based terminal kit).

1. Managers- Managers handles the core business logic of the organization .It has been purposely moved from controller to managers to enhance the reusability of the code and make the business logic available across the application as and when required through containers as a services.This classed could be extended to n numbers and supports auto loading through simple configuration available in webboot/config/appwebterminal AppTerminalWebConfig.php and AppTerminalConsoleConfig.php in web and console respectively.
2. Resources – Resources contain the assets for the web application presentation layer.Assets like CSS,JS and images
3. Views-View contain the template file having extension.php.Later when ThoughtYard will move to next level, this presentation may use .twig, .tpl, . phtml .Extensions depend upon which library an organization choose for the presentation pupose.As ThoughtYards fully supports vendor management through pacakegist.
4. Dependency injection- These classes can be instantiated using the simple configuration like managers and can be modified to handle event hooks, Inversion of control, error handlers, etc.
5. **Application and Command usage-**

Web Command to execute the output folder.

<http://thoughtyards-mikko.demo/shell/live>

Mikko Salary Structure

1. Controllers:-
2. Helpers - > (used as a dependency Injection Pattern ) these helpers are loaded as the Services in the Container. The can be called from anywhere in the program using.
3. Managers:- The Business logic is offloaded to the Managers.

Like in the Mikko application this case services of the mangers are available in the controller.

**SalaryManger**

Reason for offloading Business logic in to managers is for reusability of the Component.

Since, We also want the business logic to expose to Web along with Console.

The Managers can be available through Containers whenever and wherever required in the application.

1. The Helper
2. CSV helpers: This wehere we have created the CSV parsing. We are returning array back to the Helper and it Get generate the CSV file.

You can mention the export folder in the booting.yml file.s

Exporting the CSV to file.

The file default path to export is the “web-boot/var/export/” which can be set anything user likes inside the root folder of the application.

1. Shell Commander:-

Thanks to Pavel Tzonkov

/\*\* SHELL COMMANDER 1.0

\*

\* **@package** shcmd

\* **@version** 1.0

\* **@author** Pavel Tzonkov [pavelc@users.sourceforge.net](mailto:pavelc@users.sourceforge.net)

\* **@copyright** 2005-2007 Shell Commander Project

\* **@link** <http://sourceforge.net/projects/shcmd>

\* **@license** <http://www.opensource.org/licenses/gpl->

\* license.php

\* GPL

\*/

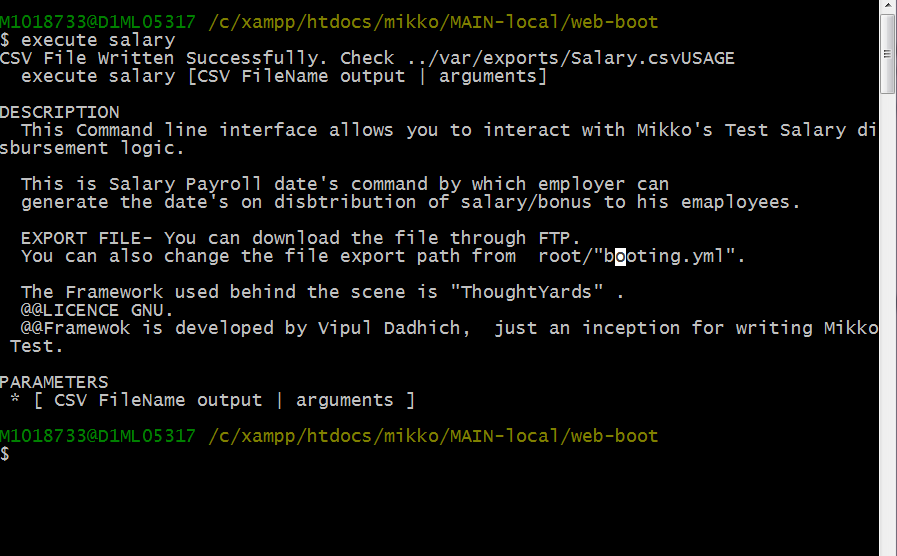
By default taken from this link and modified for the use in the application.

This Class as is downloaded added to the Application There is not reliability on this code. This Code is not modified. Just used as the dependency Injection to load the Terminal Window.

I have tried to make the program to fit in to the MVC model.

1. Console Application Usage:-

$ execute salary



PARAMETERS

[ CSV FileName output | arguments ]

$ execute salary Thougtyards.csv

SUCCESS: CSV File Written Successfully. Check ../var/exports/Thoughtyards.csv

To open the file and see what get’s print

$ bin/excel var/exports/ Thoughtyards.csv

1. **Framework(ThoughtYards)**

Under the hood, behind the scenes thoughtyards working as a spinal cord to support the application.

Framework uses light weighted YAML file for the configuration. System configurations can be played around easily by just changing values in YAML file.These small changes bring lot of internal dynamic changes in framework level configurations which is not seen by most of the frameworks.Example: Booting.Yml can control the demonstrated controlling login of the application.

booting:

logging: true

developer\_mode: true

log\_warning\_level: 2

log\_dir: var\

This is really helpful on production level application.

1. YAML - > Application supports these configurations for now.

const APP\_CONFIG='config\_%s';

const APP\_ROUTERS='router\_%s';

const APP\_VALIDATIONS='validation\_%s';

const APP\_BOOT='boot\_%s';

Usage: getConfig($type=false, $fileName=false)

**NOTE:- Do not use any “,” to separate the configurations sign in the yaml file.**

1. **Class TYCofig.php usage:-**

Booting.yml holds the configuration for the entire application:

app:

config\_dir: /web-boot/config is pointing to the configuration directory

Configurations of the application can be accesses using this method

Gateway::getAppConfig();

TYConfig::tobject is use to get the Configurations in the object form.

1. Global Registries

Provision for storing the Variable in the registry in Gateway class.

Application Components

1. Application Logs:-

*Logs directory can be specified in the booting.yml file.*

Note: This is the Framework booting configurations. Defined configurations will be available across the application by calling ThoughtyardsConfiguration Class.

*log*($msg,$level,$category='application');

generates var/ directory having file “application.log” by default.

You can also change the level of the warning logs and file for generating the error trace.

Usage:- **Gateway**::*trace*('Custom message get attached to trace',2, ‘filename’);

Sample Log:-

Loading "request" application component

Backtrace

in C:\xampp\htdocs\mikko\MAIN-local\web-boot\Gateway.php (351)in C:\xampp\htdocs\mikko\MAIN-local\vendor\thoughtyards\framework\src\Kinetics\Component\TYModule.php (391)in C:\xampp\htdocs\mikko\MAIN-local\vendor\thoughtyards\framework\src\Kinetics\Component\TYApplication.php (526)in C:\xampp\htdocs\mikko\MAIN-local\vendor\thoughtyards\framework\src\Kinetics\HttpKernel\WebKit.php (552)in C:\xampp\htdocs\mikko\MAIN-local\vendor\thoughtyards\framework\src\Kinetics\Component\CApplication.php (179)in C:\xampp\htdocs\mikko\MAIN-local\web-boot\Gateway.php (239)in C:\xampp\htdocs\mikko\MAIN-local\web-boot\index.php (17)

default logging file:-

application.log prints the Backtraces.

exception.log all ThoughtYards Exception get logged in the exception.

->Dependency Injection

ThoughtYards\Fixtures\DependencyInjection\IOC;

Usage:-

Gateway::*app*()->getContainer()->get('csv');

* RequestHandler
* ResponseHandler
* Console command runner
* GateWay –

$object= $this->getGateWayApp()->getContainer()->get('csv\_helper');

You need to define the dependency in the AppterminalConfig.php

Like

'container' => **array**(

'class' => 'TYContainer',

'services' => **array**(

'csv\_helper'=>'\TerminalKit\Payroll\Helpers\CsvHelper'

)

1. Assumption-

* User uses Windows and Linux operating systems
* This application is basically runs on the PHP CLI,
* This application runs on the framework designed and architected by Vipul i.e<Thoughtyards>
* The Docs for the Framework will be provided

1. Thanksgiving and References-

* <http://sourceforge.net/projects/shcmd/?source=dlp>
* <http://www.yiiframework.com/>
* <http://stackoverflow.com/>
* <http://php.net/>
* <http://www.thoughtyards.info/>
* <http://www.thoughtyards.com/>