ZEND Module Installer

Why

Contributors have struggled a lot in the past for having their code accepted to the OpenEMR core. Many a times the code that they produce is not within the philosophy/standards/logic of the core OpenEMR. An example was when one of the contributors had a better looking calendar version that he had wanted to contribute, it could not be included because a calendar version was already there and the new version was different in terms of platform and tools. If we had the ability to add third party modules any one who wanted to use the third party calendar could have used it while the rest of the community could continue to use the old version.

The possibility is that this module will unleash the creativity of many of the contributors. The community should not have a problem since these module will not touch the core and folks who install the core will do so at their own risk.

With a module installer OpenEMR could have multiple calendar modules, multiple billing modules, multiple lab interface, multiple impatient management modules and so on and so forth. Whenever the community feels that one module is worth adding to the core they could. This could also allow third party module installers to provide paid modules and encrypted code that is original work. This could lead to a module market place for OpenEMR.

Structure

Basic module installation requirements
Modules can be developed in two ways

1. using Zend framework
2. without using the framework

Modules, depending on their type, will be falling under
/interface/modules/zend_modules and
/interface/modules/custom_modules

- The admin folder under the /interface/modules folder contains the installation script for the modules.Running the module installer will show the list of installed and new modules from which you can register, install and activate the modules.

/interface/modules/zend_modules – For managing all the zend based module code (Zend Framework2)

config/
autoload/

- This folder will actually contain the database configuration. This is of limited use since we are avoiding this to use the default OpenEMR libraries for db access and updates.However the configurations in this files will be autoloaded based on the site configuration of the logged in user

global.php

- default db configurations except the credentials

local.php

- contains the db user credentials (The purpose of this separation is to avoid this file from getting committed in git and will remain local to the machine in which the code is running)

application.config.php

- This will contain an array of configurations for each of the modules that are installed in the system.The edits in this file will be mostly to add the new module name to the modules array

Eg: 'modules' => array(
'Application',
'ModuleName', - add your module name here
),

data/

module/ (This is where the module code will actually go in)

module1/ (Details of the folder content are explained separately)

module2/

public/

This is the default landing path. This will contain all the common files. Since different modules are being developed by different people the ideal structure of each of the sub folders in this folder should be a lib folder to contain all the re usable lib files (css,js,image) and folders with name of the modules that will contain the module specific files

css/ may contain
/lib
common css files
/module1
css files specific to module1

js/ may contain
/lib
common js files
/module1
js files specific to module1

```
        images/     may contain
                    /lib
                    Common images
                    /module1
                         Images specific to module1
       .htaccess
                 all the rewrite rules are defined here
       tests/
       vendor/
                 Right now contains zend specific library files
       composer.json
       init_autoloader
       LICENSE.txt
       index.php
```

- modified from default zend code to make the session and global variables of OpenEMR available to the modules inside the modules folder

Zend modules

Each of the modules that are created under the module folder should follow the below structure

```
   /Module1
      info.txt        - containing the module name
      table.sql     - containing additional db updates to run the module
      autoload_classmap.php
              - the autoloader configurations
      config/
         module.config.php
```

- - this will contain the module specific configuration. the default landing page in each of the module should be index(as expected by module installer). This will have configuration for controller, model and view for this module.

Module.php

- Contains the module class to handle the module behavior.
- the getServiceConfig function inside this can be used to define the table classes that are used and in future can be modified to work with zend based db operations instead of openemr code.

```
      src/
         Module1/
            /Controller
               Contains the controller files
            /Form - The forms classes should be maintained here

            /Model
```

This will contain the class files representing the tables and a Model Name Table file class where the model's db operations that can be done.The table file should be the only place where the queries are defined and we are using the

OpenEMR libraries to execute the queries. Each of the tables used can be represented as a class and a data retrieved from a function in the ModelTableClass can be converted to this object for use in controllers and views.

```
view/
    Modules1/
    ControllerClassName/
    .phtml files for all the actions defined in the controller
    layout
            This will contain the layout files for the module.
```

/interface/modules/custom_modules

This is rather a free form structure, except for some of the mandatory files.

```
ModuleName/
    index.php     - The default landing page(required)
    info.txt      - This file should contain the module name(required)
    table.sql     - Sql's to be run to get this module working(required)
    public/
    css/
      lib/           - css libraries used
      style1.css     - other css files
    js/
      lib/        - js libraries
      script1.js    - other js files
    images/
      lib/        - img libraries
      img1.jpg    - other img files
    config/       - add configuration files, if any
      lib/        - this can contain third party php libraries or php scripts
    includes/
      other php files
```

Configuration Settings

Need to enable Mod Rewrite in Apache configuration file

http://packages.zendframework.com/docs/latest/manual/en/modules/zend.view.quick-start.html#zend-view-quick-start-usage-config