TelNowEdge/FreepbxBase bundle

Version

- 2017/11/28 <0.1>: First available working version

Install

With `composer require`

Currently unavailable

With git

git clone inside composer vendor dir

```
cd /var/www/admin/libraries/Composer/vendor/telnowedge/
git clone freepbx-base
```

Update composer autoload by adding on composer.json

```
"autoload": {
  "psr-4": {
  "TelNowEdge\\FreePBX\\Base\\": "vendor/telnowedge/freepbx-base"
  }
}
```

And finally run

```
composer.phar dump-autoload
```

Overview

This FreepbxBase *bundle* provide an easy way to write FreePBX® modules like an MVC project. He works alone without any modification of FreePBX® core files except composer.json.

FreepbxBase *bundle* use Symfony® components to improve security, accessibility and support.

It register its own namespace to give access on the different components through several helpers.

FreepbxBase *bundle* introduce in FreePBX® the **Dependency Injection** concept with the Symfony® component. This component is very useful to prevent any `singleton` and share easily your object through your own code.

FreepbxBase *bundle* provide too the Symfony® **Form** component to validate your form on the server side before to save it on your sql storage.

Before start using it, you need to understand namespace and the Symfony base development concepts.

Coding standard

To check the `coding standard` please include on your module GrumPHP.

1. Require packages

```
$ composer require --dev "phpro/grumphp" "nikic/php-parser" "friendsofphp/php-cs-fixer"
```

1. Create config file for `php-cs-fixer ./php_cs`

```
<?php

return PhpCsFixer\Config::create()
    ->setRiskyAllowed(true)
    ->setRules([
        '@Symfony' => true,
        '@Symfony:risky' => true,
        'array_syntax' => true,
        'combine_consecutive_unsets' => true,
        'no_useless_else' => true,
        'no_useless_return' => true,
        'ordered_class_elements' => true,
        'ordered_imports' => true,
        'php_unit_strict' => true,
        'strict_comparison' => true,
        'strict_param' => true,
    ])
    ->setFinder(PhpCsFixer\Finder::create()
        ->exclude('vendor')
        ->in(__DIR__)
    )
;
```

2. Create GrumPHP config file `./grumphp.yml`

```
parameters:
  git_dir: .
  bin_dir: ./vendor/bin
  tasks:
    jsonlint: ~
    phpcsfixer2:
      config: "./.php_cs"
```

```
  allow_risky: true
  # rules:
  #   - "@@Symfony"
  #   - "@@Symfony:risky"
  #   - array_syntax
  #   - combine_consecutive_unsets
  #   - no_extra_consecutive_blank_lines
  #   - no_useless_else
  #   - no_useless_return
  #   - ordered_class_elements
  #   - ordered_imports
  #   - php_unit_strict
  #   - psr4
  #   - strict_comparison
  #   - strict_param
  using_cache: false
  config_contains_finder: false
phplint: ~
phpmd:
  ruleset: ['unusedcode', 'codesize']
phpparser: ~
phpspec: ~
shell: ~
xmllint: ~
yamllint:
  parse_custom_tags: true
xdebugparse: ~
```

Included components

1. doctrine/annotations
2. doctrine/cache
3. symfony/config
4. symfony/dependency-injection
5. symfony/form
6. symfony/http-foundation
7. symfony/security-csrf
8. symfony/twig-bridge
9. symfony/validator
10. symfony/yaml
11. symfony/serializer

How to use

Start a new FreePBX® module

Start a new FreePBX® module like FreePBX® practices and change only the `extends` class to `TelNowEdge\FreePBX\Base\Module\Module`.

```
<?php

namespace FreePBX\modules;

use TelNowEdge\FreePBX\Base\Module\Module;

class Foo extends Module implements \BMO
{

}
```

This extends start and bridge all Symfony® components and register a new namespace. Now you can use a PSR4 namespace inside your module.

The new register namespace is `\TelNowEdge\Module`. He is registered with `./modules` base directory. So now you can use `\TelNowEdge\Module\foo` namespace.

Note: > Take care with the case of your module name. Your class can be Foo.class.php but the folder is ./modules/foo. So the namespace is \TelNowEdge\Module\foo.

Use FreePBX® class like an entry point

Your `Foo.class.php` is the first file for FreePBX®. Now use it to call your logic Controller.

```
<?php

namespace FreePBX\modules;

use TelNowEdge\FreePBX\Base\Module\Module;
use TelNowEdge\Module\foo\Controller\FooBarController;

class Foo extends Module implements \BMO
{
    public function install()
    {
        $this
            ->get('TelNowEdge\Module\foo\Resources\Migrations\TableMigration')
```

```
            ->migrate()
            ;
    }

    public static function myGuiHooks()
    {
        return array('core');
    }

    public function doGuiHook(&$cc)
    {
        $this
            ->processDeviceGui($cc)
            ;
    }

    private function processDeviceGui(&$cc)
    {
        $request = $this->get('request');

        if ('devices' === $request->query->get('display')) {
            if (true === $request->isMethod('POST')) {
                if ('edit' === $request->request->get('action')) {
                    $this->get(FooBarController::class)
                         ->updateAction($request, $cc)
                        ;
                }

                if ('add' === $request->request->get('action')) {
                    $this->get(FooBarController::class)
                         ->createAction($request, $cc)
                        ;
                }
            } else {
                $this->get(FooBarController::class)
                     ->showAction($request, $cc)
                    ;
            }
        }
    }
}
```

Your controller `./modules/foo/Controller/FooBarController.php`

```
<?php

namespace TelNowEdge\Module\foo\Controller;
```

```php
use TelNowEdge\FreePBX\Base\Controller\AbstractController;

class FooBarController extends AbstractController
{
    [...]
}
```

Reference

Module

Entry point to start FreepbxBase *bundle*.

FreePBX® module must extends `TelNowEdge\FreePBX\Base\Module\Module`

Controller

- Symfony Documentation

Your Controller must extends `TelNowEdge\FreePBX\Base\Controller\AbstractController`

```php
protected $container;

FormFactory function createForm(FormInterface $type, $data = null, array $options = array()

string function render(string $templatePath, array $data = array());

mixed function get(string $service);
```

1. createForm()
2. render()

   Render return the compile html from the template. Append on FreePBX® with the FreePBX® practices.

   ```php
   $html = $this->render('foo.html.twig', array(
       'form' => $form->createView(),
   ));

   $cc->addguielem(_('Foo'), new \gui_html('Foo', $html), 1, null, _('TelNowEdge'));
   ```

3. get()

Model

- Symfony Documentation

Note: > This bundle don't use Doctrine ORM. But the way is the same.

`Model` is the Database representation. This class must not extends anything.

On each properties, you can add a validator.

```
<?php

namespace TelNowEdge\Module\foo\Model;

use Symfony\Component\Validator\Constraints as Assert;

class Foo
{
    protected $id;

    /**
     * @Assert\NotBlank()
     */
    protected $name;

    public function getId()
    {
        return $this->id;
    }

    public function setId($id)
    {
        $this->id = $id;

        return $this;
    }

    public function getName()
    {
        return $this->name;
    }

    public function setName($name)
    {
        $this->name = $name;

        return $this;
```

```
    }
}
```

Repository

`Repository` get informations from sql storage and map with Model. ORM like very lite.

Your Repository must extends `TelNowEdge\FreePBX\Base\Repository\AbtractRepository`

```
array sqlToArray(array $sqlRes);

\Doctrine\DBAL\Statement fetch(\Doctrine\DBAL\Statement $stmt);

\Doctrine\DBAL\Statement fetchAll(\Doctrine\DBAL\Statement $stmt);

Model objectFromArray(string $modelClass, array $sqlToArrayRes);
```

1. sqlToArray() Transform sql results set to an array for `objectFromArray()`

   `sqlToArray()` need a formatted input.

   ```
   SELECT t.id t__id, t.name t__name, t.long_name t__long_name, t2.id t2__id
   FROM table t INNER JOIN table2 t2 ON (t2.id = t1.id)
   ```

   `sqlToArray()` return an associative array like:

   ```
   array(
       't' => array('id' => '1', 'name' => 'foo', 'longName' => 'foobar'),
       't2' => array('id' => 1)
   )
   ```

   Note: > The ___ was remove to create table key and _ was camel case.

2. objectFromArray() Map the `sqlToArray()` to the model. On each properties, he try to call the setter.

   ```
   private function mapModel(array $res)
   {
       $foo = $this->objectFromArray(Foo::class, $res['t']);
       $fooBar = $this->objectFromArray(FooBar::class, $res['t2']);

       return $foo->setFooBar($fooBar);
   }
   ```

DbHandler

`DbHandler` save data from the `Model` to the sql.

Your Repository must extends `TelNowEdge\FreePBX\Base\Handler\AbtractDbHandler`

```php
<?php

namespace TelNowEdge\Module\foo\Handler\DbHandler;

use TelNowEdge\FreePBX\Base\Handler\AbstractDbHandler;
use TelNowEdge\Module\foo\Model\Foo;

class PhoneProvisionDbHandler extends AbstractDbHandler
{
    public function create(Foo $foo)
    {
        $sql = "INSERT INTO Foo (`id`, `name`, `value`) VALUES (:id, :name, :value)";
        $stmt = $this->connection->prepare($sql);
        $stmt->bindParam('id', $foo->getId());
        $stmt->bindParam('name', $foo->getName());
        $stmt->bindParam('value', $foo->getValue());

        $stmt->execute();
    }
```

Form

`Form` provide an easy way to build and validate your form.

This component is used exactly like Symfony does.

- Symfony documentation
- Advanced documentation

Validator

`Validator` works with `Form` to validate it on server side.

This component is used exactly like Symfony does.

- Symfony documentation
- Advanced documentation

Dependency Injection

`Dependency Injection` create a container of services to deal with on your code.

This component is used exactly like Symfony does.

- Symfony documentation

Twig

`Twig` is a templating component. Cery useful to render the Symfony® forms

This component is used exactly like Symfony does.

- Symfony documentation

Todo

1. Increase security in service.yml with publc / private service
2. Create an Acme module