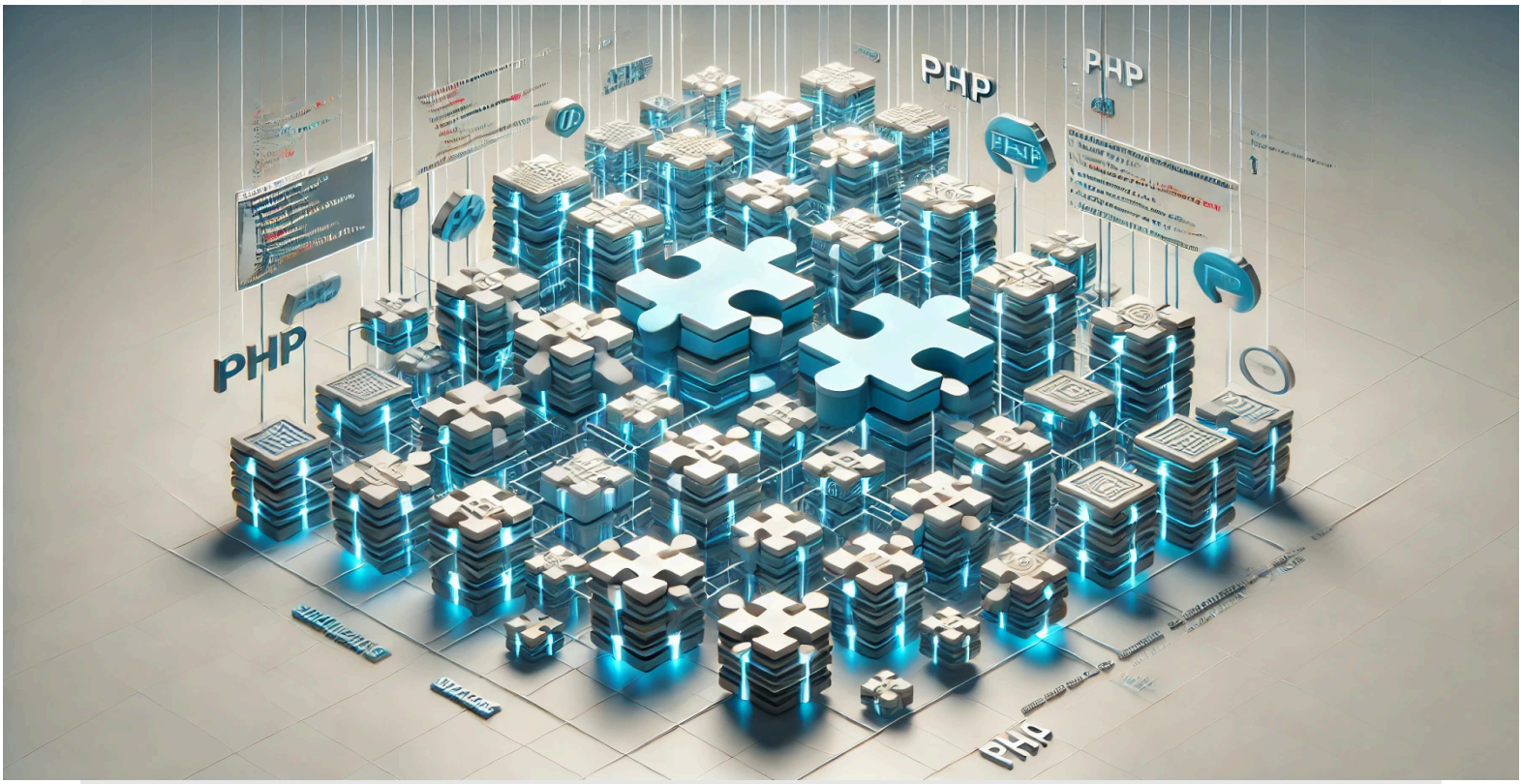
TABLE OF CONTENTS



PARLONS-EN – CONSTRUIRE UN FRAMEWORK PHP MODULAIRE, PARTIE 2

Auteur(s) : Louis Ouellet

Il est temps pour la partie 2 ! Dans la partie précédente, nous avons mis en place la base de notre framework PHP modulaire. Cette fois-ci, nous allons nous concentrer sur l'expansion de ses capacités pour prendre en charge les objectifs suivants :

- Ajouter la prise en charge d'extensions
- Commencer à mettre en œuvre un module en ligne de commande (CLI)
- Ajouter la prise en charge de modèles (models) pour créer des méthodes partagées nécessitant une base de données
- Ajouter la prise en charge de helpers pour créer des méthodes partagées ne nécessitant pas de base de données

Ces améliorations nous offriront la flexibilité dont nous avons besoin pour construire des applications modulaires, maintenables et extensibles. Passons en revue chaque mise à jour

étape par étape.

MISE À JOUR DU BOOTSTRAP

Nous allons commencer par mettre à jour la classe `Bootstrap` afin d'inclure les helpers et les models. Nous avons également changé la classe par défaut de `CLI` pour `LaswitchTech\Core\CLI` afin de pouvoir commencer à y travailler avant de l'exporter dans son propre module.

Source: Bootstrap.php

src/Bootstrap.php

```php
<?php

/**
 * Core Framework - Bootstrap
 *
 * @license    MIT (https://mit-license.org/)
 * @author     Louis Ouellet <louis@laswitchtech.com>
 */

// Déclaration du namespace
namespace LaswitchTech\Core;

// Import de classes supplémentaires dans l'espace de noms global
use LaswitchTech\Core\Config;
use LaswitchTech\Core\Strap;
use Exception;

class Bootstrap {

    const Default = [
        "LOG" => [
            "class" => "\LaswitchTech\Core\Log",
            "scope" => [
                "Router",
                "API",
```

```
                "CLI"
            ]
        ],
        "REQUEST" => [
            "class" => "\LaswitchTech\Core\Request",
            "scope" => [
                "Router",
                "API",
                "CLI"
            ]
        ],
        "OUTPUT" => [
            "class" => "\LaswitchTech\Core\Output",
            "scope" => [
                "Router",
                "API",
                "CLI"
            ]
        ],
        "NET" => [
            "class" => "\LaswitchTech\coreNet\Net",
            "scope" => [
                "Router",
                "API",
                "CLI"
            ]
        ],
        "HELPER" => [
            "class" => "\LaswitchTech\Core\Helpers",
            "scope" => [
                "Router",
                "API",
                "CLI"
            ]
        ],
        "DATABASE" => [
            "class" => "\LaswitchTech\coreDatabase\Database",
            "scope" => [
                "Router",
                "API",
                "CLI"
            ]
        ],
```

```
        "MODEL" => [
            "class" => "\LaswitchTech\Core\Models",
            "scope" => [
                "Router",
                "API",
                "CLI"
            ]
        ],
        "LOCALE" => [
            "class" => "\LaswitchTech\coreLocale\Locale",
            "scope" => [
                "Router",
                "API",
                "CLI"
            ]
        ],
        "ENCRYPTION" => [
            "class" => "\LaswitchTech\coreEncryption\Encryption",
            "scope" => []
        ],
        "CSRF" => [
            "class" => "\LaswitchTech\coreCSRF\CSRF",
            "scope" => [
                "Router",
                "API"
            ]
        ],
        "SLS" => [
            "class" => "\LaswitchTech\coreSLS\SLS",
            "scope" => [
                "Router",
                "API",
                "CLI"
            ]
        ],
        "SMS" => [
            "class" => "\LaswitchTech\coreSMS\SMS",
            "scope" => [
                "Router",
                "API",
                "CLI"
            ]
```

```
        ],
        "SMTP" => [
            "class" => "\LaswitchTech\coreSMTP\SMTP",
            "scope" => [
                "Router",
                "API",
                "CLI"
            ]
        ],
        "IMAP" => [
            "class" => "\LaswitchTech\coreIMAP\IMAP",
            "scope" => [
                "Router",
                "API",
                "CLI"
            ]
        ],
        "INSTALLER" => [
            "class" => "\LaswitchTech\coreInstaller\Installer",
            "scope" => [
                "Router",
                "API",
                "CLI"
            ]
        ],
        "AUTH" => [
            "class" => "\LaswitchTech\coreAuth\Auth",
            "scope" => [
                "Router",
                "API"
            ]
        ],
        "ROUTER" => [
            "class" => "\LaswitchTech\coreRouter\Router",
            "scope" => [
                "Router"
            ]
        ],
        "API" => [
            "class" => "\LaswitchTech\coreAPI\API",
            "scope" => [
                "API"
            ]
```

```
        ],
        "CLI" => [
            "class" => "\LaswitchTech\Core\CLI",
            "scope" => [
                "CLI"
            ]
        ]
    ];

    /**
     * Constructeur
     */
    public function __construct($scope){

        // Définir la variable globale
        global $CONFIG;

        // Initialiser Config
        $CONFIG = new Config(['bootstrap']);

        // Récupérer les « straps »
        $straps = $CONFIG->get('bootstrap');

        // Boucler à travers les straps
        foreach(self::Default as $strap => $config){

            // Vérifier s'il existe un strap alternatif
            if(isset($straps[$strap])){

                // Boucler à travers le strap alternatif
                foreach($config as $key => $value){

                    // Vérifier si le strap alternatif a la clé
                    if(isset($straps[$strap][$key])){

                        // Définir le strap alternatif
                        $config[$key] = $straps[$strap][$key];
                    }
                }
            }

            // Vérifier si le strap ne fait pas partie du scope
```

```
            if(!in_array($scope,$config['scope'])) continue;

            // Initialiser la variable globale
            global ${$strap};

            // Définir la classe
            $class = $config['class'];

            // Vérifier si la classe existe
            if(class_exists($class)){

                // Initialiser la classe dans l'espace de noms global
                ${$strap} = new $class();
            } else {

                // Définir la valeur par défaut comme null
                ${$strap} = new Strap();
            }
        }
    }
}
```

MISE À JOUR DE LA CLASSE REQUEST

Ensuite, mettons à jour la classe `Request` afin qu'elle puisse gérer les arguments de la ligne de commande. Dans le constructeur, nous stockons la variable globale `$argv` dans une propriété `Arguments` (avec un tableau vide par défaut).

Source: Request.php

src/Request.php

```
<?php

/**
 * Core Framework - Request
 *
```

```
 * @license    MIT (https://mit-license.org/)
 * @author     Louis Ouellet <louis@laswitchtech.com>
 */

// Déclaration du namespace
namespace LaswitchTech\Core;

// Import de classes supplémentaires dans l'espace de noms global
use Exception;

class Request {

    // Propriétés
    private $Get;
    private $Post;
    private $Files;
    private $Server;
    private $Cookie;
    private $Request;
    private $Arguments;

    /**
     * Constructeur
     */
    public function __construct(){

        // Importer les variables globales
        global $_GET, $_POST, $_FILES, $_SERVER, $_COOKIE, $_REQUEST,
$argv;

        // Définir les propriétés
        $this->Get = $_GET;
        $this->Post = $_POST;
        $this->Files = $_FILES;
        $this->Server = $_SERVER;
        $this->Cookie = $_COOKIE;
        $this->Request = $_REQUEST;
        $this->Arguments = $argv ?? [];

        // Nettoyer les variables globales
        unset($_SERVER, $_GET, $_POST, $_FILES, $_COOKIE, $_REQUEST,
$argv);
```

```
    }

    /**
     * Récupérer l'hôte
     */
    public function getHost() {
        return $this->Server['HTTP_HOST'] ?? '';
    }

    /**
     * Récupérer le préfixe SSL
     */
    public function getHostSSL() {
        return $this->Server['HTTPS'] == 'on' ? 'https://' : 'http://';
    }

    /**
     * Récupérer l'adresse complète de l'hôte
     */
    public function getHostAddress() {
        return defined('STDIN') ? 'localhost' : $this->getHostSSL() .
$this->getHost();
    }

    /**
     * Récupérer l'URI
     * @return string
     */
    public function getUri() {
        return parse_url($this->Server['REQUEST_URI'] ?? '',
PHP_URL_PATH);
    }

    /**
     * Récupérer les segments de l'URI
     * @return array
     */
    public function getUriSegments() {
        return array_filter(explode( '/', $this->getUri()));
    }

    /**
     * Récupérer et retourner la propriété Namespace
```

```
     */
    public function getNamespace(){
        $namespace = "";
        foreach($this->getUriSegments() as $Segment){
            $namespace .= "/{$Segment}";
        };
        return $namespace;
    }

    /**
     * Récupérer la méthode de requête
     * @return string
     */
    public function getMethod() {
        return $this->Server["REQUEST_METHOD"] ?? 'GET';
    }

    /**
     * Récupérer la chaîne de requête
     * @return string
     */
    public function getQueryString() {
        return $this->Server['QUERY_STRING'] ?? '';
    }

    /**
     * Récupérer les paramètres
     * @param string $type
     * @param string $key
     * @return mixed
     */
    public function getParams($type, $key = null){
        switch(strtoupper($type)){
            case 'GET':
                $array = $this->Get;
                break;
            case 'POST':
                $array = $this->Post;
                break;
            case 'FILES':
                $array = $this->Files;
                break;
```

```
            case 'COOKIE':
                $array = $this->Cookie;
                break;
            case 'SERVER':
                $array = $this->Server;
                break;
            case 'QUERY':
                parse_str($this->Server['QUERY_STRING'], $array);
                break;
            case 'REQUEST':
                $array = $this->Request;
                break;
            default:
                return null;
        }
        return !is_null($key) ? ($array[$key] ?? null) : $array;
    }

    /**
     * Décoder les données de REQUEST
     * @param string $string
     * @return string
     */
    public function decode($string){
        return urldecode(base64_decode($string));
    }

    /**
     * Récupérer les arguments
     * @param int $index
     * @return mixed
     */
    public function getArguments($index = null){
        return !is_null($index) ? ($this->Arguments[$index] ?? null) :
$this->Arguments;
    }

    /**
     * Demander une entrée utilisateur
     * @param string $string
     * @param array|int|string $options
     * @param string $default
     * @return string
```

```
     */
    public function request($string, $options = null, $default = null){
        if(defined('STDIN')){
            $modes = ['select','text','string'];
            $mode = 'string';
            if($options != null || $options == 0){
                if(is_array($options)){
                    $mode = 'select';
                } else if(is_int($options)){
                    $mode = 'text';
                } else {
                    if(is_string($options)){ $default = $options; }
                }
            }
            $stdin = function(){
                $handle = fopen ("php://stdin","r");
                return str_replace("\n",'',fgets($handle));
            };
            switch($mode){
                case"select":
                    $answer = null;
                    foreach($options as $key => $value){
                        $options[$key] = strtoupper($value);
                    }
                    while($answer == null ||
!in_array(strtoupper($answer),$options)){
                        print_r($string . ' (');
                        foreach($options as $key => $option){
                            if($key > 0){ print_r('/'); }
                            print_r($option);
                        }
                        print_r(')');
                        if($default != null){ print_r('['.$default.']');
}
                        print_r(': ');
                        $answer = $stdin();
                        if($default != null && $answer == ""){ $answer =
$default; }
                    }
                    break;
                case"text":
                    $answer = '';
```

```
                    $exits = ['END','EXIT','QUIT','EOF',':Q',''];
                    $count = 0;
                    $max = 5;
                    $print = false;
                    if(is_bool($default)){ $print = $default; }
                    if(is_int($options)){ $max = $options; }
                    if($print){
                        print_r($string . ' tapez (');
                        foreach($exits as $key => $exit){
                            if($key > 0){ print_r('/'); }
                            print_r($exit);
                        }
                        print_r(') pour sortir' . PHP_EOL);
                    } else {
                        print_r($string . PHP_EOL);
                    }
                    do {
                        $line = fgets(STDIN);
if(in_array(strtoupper(str_replace("\n",'',$line)),$exits)){
                            if($max <= 0){ $max = 1; }
                            $count = $max;
                        } else { $answer .= $line; $count++; }
                    } while ($count < $max || $max <= 0);
                    break;
                default:
                    $answer = null;
                    while($answer == null){
                        print_r($string . ' ');
                        if($default != null){ print_r('['.$default.']');
}
                        print_r(': ');
                        $answer = $stdin();
                        if($default != null && $answer == ""){ $answer =
$default; }
                        if($answer == ''){ $answer = null; }
                    }
                    break;
            }
            $answer = trim($answer,"\n");
            if($answer == ''){ $answer = null; }
            return $answer;
        }
    }
```

```
}
```

Nous ajoutons ensuite une méthode `getArguments()` pour récupérer les arguments et déplaçons toutes les méthodes de demande d'entrée utilisateur (auparavant dans `coreCLI`) vers cette classe.

MISE À JOUR DE LA CLASSE OUTPUT

Nous devons également mettre à jour la classe `Output` afin de gérer à la fois la sortie dans le navigateur et en ligne de commande. Voici un exemple de détection CLI (`STDIN`) et de messages colorés.

Source: Output.php

src/Output.php

```php
<?php

/**
 * Core Framework - Output
 *
 * @license    MIT (https://mit-license.org/)
 * @author     Louis Ouellet
 */

// Déclaration du namespace
namespace LaswitchTech\Core;

// Import de classes supplémentaires dans l'espace de noms global
use Exception;

class Output {

    // Propriétés
    protected $Colors = [
```

```
        "default" => "3[39m",
        "black" => "3[30m",
        "red" => "3[31m",
        "green" => "3[32m",
        "yellow" => "3[33m",
        "blue" => "3[34m",
        "magenta" => "3[35m",
        "cyan" => "3[36m",
        "light-gray" => "3[37m",
        "dark-gray" => "3[90m",
        "light-red" => "3[91m",
        "light-green" => "3[92m",
        "light-yellow" => "3[93m",
        "light-blue" => "3[94m",
        "light-magenta" => "3[95m",
        "light-cyan" => "3[96m",
        "white" => "3[97m",
    ];

    /**
     * Constructeur
     */
    public function __construct(){}

    /**
     * Affiche une chaîne
     */
    public function print($string, $httpHeaders=array()) {

        // Vérifier si le script s'exécute en mode CLI
        if(defined('STDIN')){
            // Affichage dans la console
            print_r($string . PHP_EOL);
        } else {
            // Sinon, gérer la sortie pour le navigateur et les en-têtes
            if (!headers_sent()) {
                header_remove('Set-Cookie');
                if (is_array($httpHeaders) && count($httpHeaders)) {
                    foreach ($httpHeaders as $httpHeader) {
                        header($httpHeader);
                    }
                }
                if(is_array($string) || is_object($string)){
```

```
                    $string = json_encode($string,
JSON_UNESCAPED_SLASHES | JSON_PRETTY_PRINT);
                }
                echo $string;
                exit;
            }
        }
    }

    /**
     * Affiche une chaîne colorée
     */
    public function set($string, $color = 'default'){
        if(defined('STDIN') && isset($this->Colors[$color])){
            return $this->Colors[$color] . $string .
$this->Colors['default'];
        } else {
            return $string;
        }
    }

    public function error($string) {
        $this->print($this->set($string, 'red'));
    }

    public function success($string) {
        $this->print($this->set($string, 'green'));
    }

    public function warning($string) {
        $this->print($this->set($string, 'yellow'));
    }

    public function info($string) {
        $this->print($this->set($string, 'cyan'));
    }

    /**
     * Affiche l'aide en ligne de commande
     */
    public function help($string = null) {
```

```
        // Vérifier si le script s'exécute en mode CLI
        if(!defined('STDIN')){ return; }

        // Récupérer les variables globales
        global $REQUEST, $CONFIG;

        // Récupérer les arguments
        $arguments = $REQUEST->getArguments();

        // Variables initiales
        $file = $arguments[0] ?? null;
        $command = $arguments[1] ?? null;
        $action = $arguments[2] ?? null;

        // Si une chaîne est fournie, l'afficher
        if($string){ $this->print($string); }

        // Tenter de localiser une commande spécifique
        if($command){
            $path = $CONFIG->root() . "/Command/" . ucfirst($command) .
"Command" . ".php";
            if(!is_file($path)){
                $path = $CONFIG->root() . "/lib/plugins/" .
ucfirst($command) . "/Command.php";
                if(!is_file($path)){
                    $command = null;
                }
            }
        }
        if($command){
            if(!class_exists(ucfirst($command) . "Command")){
                require_once $path;
                if(!class_exists(ucfirst($command) . "Command")){
                    $command = null;
                }
            }
        }

        if($command){
            // Si une commande est trouvée, afficher l'utilisation pour
cette commande
            $class = ucfirst($command) . "Command";
            $class = new $class();
```

```
            if(!method_exists($class, $action . "Action")){
                $action = null;
            }

            $this->print("Usage: $file $command [action] [options]");
            $this->print("Actions disponibles :");
            foreach(get_class_methods($class) as $method){
                if(substr($method,-6) == 'Action'){
                    $this->print(" - " .
strtolower(str_replace('Action','',$method)));
                }
            }
        } else {
            // Aide générale
            $this->print("Usage: $file [command] [action] [options]");
            $this->print("Commandes disponibles :");

            // Depuis /Command/
            foreach(scandir($CONFIG->root() . "/Command/") as $command){
                if(str_contains($command, 'Command.php')){
                    $this->print(" - " .
strtolower(str_replace('Command.php','',$command)));
                }
            }

            // Depuis /lib/plugins/
            foreach(scandir($CONFIG->root() . "/lib/plugins/") as
$command){
                if(is_file($CONFIG->root() . "/lib/plugins/" . $command
. "/Command.php")){
                    $this->print(" - " . strtolower($command));
                }
            }
        }

        // Arrêter l'exécution
        exit();
    }
}
```

MISE EN PLACE DES HELPERS

Les helpers fournissent de petites méthodes réutilisables. Pour cela, nous créons d'abord une classe de base `Helper` qui pourra être étendue si nous avons besoin de propriétés ou de méthodes communes à l'avenir.

Source: Helper.php

src/Helper.php

```php
<?php

/**
 * Core Framework - Helper
 *
 * @license    MIT (https://mit-license.org/)
 * @author     Louis Ouellet
 */

// Déclaration du namespace
namespace LaswitchTech\Core;

class Helper {
    // Pour l'instant vide, mais peut contenir une logique commune pour les helpers
}
```

Ensuite, nous implémentons la classe principale `Helpers` qui charge dynamiquement les fichiers helper individuels depuis votre application et depuis les plugins.

Source: Helpers.php

src/Helpers.php

```php
<?php
```

```php
/**
 * Core Framework - Helpers
 *
 * @license    MIT (https://mit-license.org/)
 * @author     Louis Ouellet
 */

// Déclaration du namespace
namespace LaswitchTech\Core;

// Import de classes supplémentaires dans l'espace de noms global
use Exception;

class Helpers {

    // Propriétés
    protected $Path = null;
    protected array $Helpers = []; // stockage ici

    public function __construct() {

        // Importer les variables globales
        global $CONFIG;

        // Charger les Helpers de l'application
        $this->Path = $CONFIG->root() . "/Helper";
        if(is_dir($this->Path)){
            foreach(scandir($this->Path) as $helper){
                if (preg_match('/(.+)Helper\.php$/i', $helper,
$matches)) {
                    require_once $this->Path . "/" . $helper;
                    $baseName = $matches[1];
                    $className = $baseName . 'Helper';
                    if (class_exists($className)) {
                        $this->Helpers[$baseName] = new $className();
                    }
                }
            }
        }

        // Charger les Helpers des plugins
```

```
        $this->Path = $CONFIG->root() . "/lib/plugins";
        if(is_dir($this->Path)){
            foreach(scandir($this->Path) as $plugin){
                // Ne charger que s'il n'existe pas déjà un helper avec
ce nom
                if(!isset($this->Helpers[ucfirst($plugin)])){
                    $path = $this->Path . "/" . $plugin . "/Helper.php";
                    if(is_file($path)){
                        $baseName = ucfirst($plugin);
                        $className = $baseName . 'Helper';
                        // Tenter de charger la classe si elle existe
                        if (class_exists($className)) {
                            $this->Helpers[$baseName] = new
$className();
                        }
                    }
                }
            }
        }

    // Méthodes magiques pour getters/setters
    public function __get($name) {
        return $this->Helpers[$name] ?? null;
    }

    public function __set($name, $value) {
        $this->Helpers[$name] = $value;
    }
}
```

MISE EN PLACE DES MODELS

Les models fonctionnent de manière similaire aux Helpers, mais nécessitent généralement une connexion à la base de données. Nous stockons une référence à l'objet global `$DATABASE` dans la classe de base `Model`.

Source: Modal.php

src/Modal.php

```php
<?php

/**
 * Core Framework - Model
 *
 * @license    MIT (https://mit-license.org/)
 * @author     Louis Ouellet
 */

namespace LaswitchTech\Core;

use Exception;

class Model {

    // Propriétés
    protected $Database;

    public function __construct(){
        // Récupérer la base de données globale
        global $DATABASE;
        $this->Database = $DATABASE;
    }
}
```

Tout comme les Helpers, la classe `Models` charge dynamiquement tous les modèles depuis votre dossier `/Model` et les répertoires de plugins :

Source: Modals.php

src/Modals.php

```php
<?php

/**
 * Core Framework - Models
```

```
 *
 * @license    MIT (https://mit-license.org/)
 * @author     Louis Ouellet
 */

namespace LaswitchTech\Core;

use Exception;

class Models {

    protected $Path = null;
    protected array $Models = [];

    public function __construct() {

        global $CONFIG;

        // Charger les Models de l'application
        $this->Path = $CONFIG->root() . "/Model";
        if(is_dir($this->Path)){
            foreach(scandir($this->Path) as $model){
                if (preg_match('/(.+)Model\.php$/i', $model, $matches))
{
                    require_once $this->Path . "/" . $model;
                    $baseName = $matches[1];
                    $className = $baseName . 'Model';
                    if (class_exists($className)) {
                        $this->Models[$baseName] = new $className();
                    }
                }
            }
        }

        // Charger les Models des plugins
        $this->Path = $CONFIG->root() . "/lib/plugins";
        if(is_dir($this->Path)){
            foreach(scandir($this->Path) as $plugin){
                if(!isset($this->Models[ucfirst($plugin)])){
                    $path = $this->Path . "/" . $plugin . "/Model.php";
                    if(is_file($path)){
                        $baseName = ucfirst($plugin);
                        $className = $baseName . 'Model';
```

```
                        if (class_exists($className)) {
                            $this->Models[$baseName] = new $className();
                        }
                    }
                }
            }
        }

    public function __get($name) {
        return $this->Models[$name] ?? null;
    }

    public function __set($name, $value) {
        $this->Models[$name] = $value;
    }
}
```

MISE EN PLACE DU MODULE COMMAND-LINE

Pour mettre en place une interface en ligne de commande (CLI), nous allons créer une classe `Command` que les développeurs pourront étendre. Ensuite, nous ajouterons une classe `CLI` qui identifiera et exécutera les commandes et les actions.

CLASSE DE BASE COMMAND

Source: Command.php

src/Command.php

```
<?php

/**
 * Core Framework - Command
```

```
 *
 * @license    MIT (https://mit-license.org/)
 * @author     Louis Ouellet
 */

namespace LaswitchTech\Core;

use Exception;

class Command {

    // Propriétés globales
    protected $Model;
    protected $Helper;
    protected $Output;
    protected $Request;

    public function __construct(){
        global $MODEL, $HELPER, $OUTPUT, $REQUEST;
        $this->Model = $MODEL;
        $this->Helper = $HELPER;
        $this->Output = $OUTPUT;
        $this->Request = $REQUEST;
    }

    /**
     * Méthode magique pour intercepter les méthodes non définies
     */
    public function __call($name, $arguments) {
        $this->Output->help();
    }
}
```

CLASSE CLI

Source: CLI.php

src/CLI.php

```php
<?php

/**
 * Core Framework - CLI
 *
 * @license    MIT (https://mit-license.org/)
 * @author     Louis Ouellet
 */

namespace LaswitchTech\Core;

use Exception;

class CLI {

    protected $Output;
    protected $Request;
    protected $Config;

    protected $Arguments;
    protected $Class;
    protected $Path;
    protected $Command;
    protected $Action;

    public function __construct(){
        global $OUTPUT, $REQUEST, $CONFIG;
        $this->Output = $OUTPUT;
        $this->Request = $REQUEST;
        $this->Config = $CONFIG;
        $this->Arguments = $this->Request->getArguments();
    }

    public function run(){
        // Analyser les entrées standard
        if(count($this->Arguments) > 0){
            // Le premier argument est le nom du script
            unset($this->Arguments[0]);

            if(count($this->Arguments) > 0){
                // Identifier la Commande
                $this->Command = ucfirst($this->Arguments[1] .
```

```
"Command");
                unset($this->Arguments[1]);

                // Vérifier le chemin dans /Command/ et /lib/plugins/
                $this->Path = $this->Config->root() . "/Command/" .
$this->Command . ".php";
                if(!is_file($this->Path)){
                    $this->Path = $this->Config->root() .
"/lib/plugins/" . strtolower(str_replace('Command','',$this->Command)) .
"/Command.php";
                }

                if(is_file($this->Path)){
                    require_once $this->Path;
                    if(class_exists($this->Command)){
                        $this->Class = new $this->Command();

                        if(count($this->Arguments) > 0){
                            $this->Action = $this->Arguments[2] .
"Action";
                            unset($this->Arguments[2]);

                            if(method_exists($this->Class,
$this->Action)){
                                $this->Class->{$this->Action}();
                            } else {
                                $this->Output->help("L'action " .
strtolower(str_replace('Action','',$this->Action)) . " n'est pas
disponible");
                            }
                        } else {
                            $this->Output->help();
                        }
                    } else {
                        $this->Output->help("Impossible d'initialiser la
commande " . strtolower(str_replace('Command','',$this->Command)));
                    }
                } else {
                    $this->Output->help("La commande " .
strtolower(str_replace('Command','',$this->Command)) . " n'est pas
disponible");
                }
            } else {
```

```
                $this->Output->help();
            }
        } else {
            $this->Output->help("Erreur Interne");
        }
    }
}
```

TESTER LE CODE : CRÉATION D'UN PLUGIN "DEBUG"

Créons un plugin simple nommé `debug` dans `/lib/plugins/debug/Command.php`. Cela démontre comment définir des commandes personnalisées prises en charge par notre nouvelle infrastructure CLI :

Source: Command.php

lib/plugins/debug/Command.php

```
<?php

/**
 * Core Framework - DebugCommand
 *
 * @license    MIT (https://mit-license.org/)
 * @author     Louis Ouellet
 */

use LaswitchTech\Core\Command;

class DebugCommand extends Command {

    public function __construct(){
        parent::__construct();
    }

    public function executeAction(){
```

```
        $this->Output->print("Debugging...");
    }
}
```

EXÉCUTER LA CLI

Voici un script d'exemple `debug.php` dans un dossier `tmp/` pour tester la CLI. Rendez-le exécutable (``chmod +x tmp/debug.php``) et lancez-le ainsi : ``./tmp/debug.php debug execute``

Source: debug.php

tmp/debug.php

```
#!/usr/bin/env php
<?php

use LaswitchTech\Core\Bootstrap;
use LaswitchTech\Core\CLI;

require dirname(__DIR__) . "/vendor/autoload.php";

// Initialiser Bootstrap
$BOOTSTRAP = new Bootstrap("CLI");

// Définir EOL en fonction de l'environnement
$_EOL = defined("STDIN") ? PHP_EOL : "<br>";

// Afficher quelques infos de débogage
foreach(get_defined_vars() as $key => $value){
    if(substr($key,0,1) == "_" || in_array($key,["argv","argc"]))
continue;
    echo "Variable Name: " . $key . " = " . get_class($value) . $_EOL;
}

echo $_EOL . "Host: " . $REQUEST->getHostAddress() . $_EOL;
echo $_EOL . "Parameter: " . $REQUEST->getParams("GET","query") . $_EOL;
```

```php
if(defined('STDIN') && !empty($argv)){
    echo $_EOL . "CLI: " . $_EOL;
    $CLI->run();
}

echo $_EOL . "End of Script" . $_EOL;
```

EXEMPLE DE SORTIE

Lorsque vous exécutez :

```
./tmp/debug.php debug execute
```

Vous devriez voir quelque chose comme :

```
Variable Name: BOOTSTRAP = LaswitchTech\Core\Bootstrap
Variable Name: REQUEST = LaswitchTech\Core\Request
Variable Name: CLI = LaswitchTech\Core\CLI
Variable Name: CONFIG = LaswitchTech\Core\Config
Variable Name: LOG = LaswitchTech\Core\Log
Variable Name: OUTPUT = LaswitchTech\Core\Output
Variable Name: NET = LaswitchTech\Core\Strap
Variable Name: HELPER = LaswitchTech\Core\Helpers
Variable Name: DATABASE = LaswitchTech\Core\Strap
Variable Name: MODEL = LaswitchTech\Core\Models
Variable Name: LOCALE = LaswitchTech\Core\Strap
Variable Name: SLS = LaswitchTech\Core\Strap
Variable Name: SMS = LaswitchTech\Core\Strap
Variable Name: SMTP = LaswitchTech\Core\Strap
Variable Name: IMAP = LaswitchTech\Core\Strap
Variable Name: INSTALLER = LaswitchTech\Core\Strap

Host: localhost

Parameter:
```

```
CLI:
Debugging...

End of Script
```

CONCLUSION

DÉPÔT GITHUB

Dépôt GitHub - v0.0.7

Dans cette deuxième partie de notre série, nous avons étendu notre framework PHP modulaire pour prendre en charge les opérations en ligne de commande, les classes helper et les models. En créant une classe générique `CLI` et une classe `Command` de base, nous pouvons facilement étendre le framework avec de nouvelles commandes et actions. L'ajout des `Helpers` et des `Models` offre un moyen plus propre et mieux organisé de partager du code dans votre application, ce qui facilite la maintenance et l'extension des fonctionnalités sans duplication de code.

Notre framework commence à prendre forme avec une séparation claire des responsabilités et la flexibilité d'ajouter ou de supprimer des fonctionnalités selon les besoins. Dans les parties à venir, nous continuerons à peaufiner notre framework, éventuellement en intégrant une journalisation plus robuste, une gestion de configuration plus avancée et d'autres modules qui pourront encore simplifier le développement.

TAGSGENERALCORE-FRAMEWORK-FRFRAMEWORKCOREMODULARITYMODULARMODULELIGHTWEIGHTAPPLICATIONPHPCAKEPHPSYMFONYLEARNLEANINGMAINTAINABILITYMAINTAIN

- Twitter
- Facebook

- LinkedIn
- Reddit
- Telegram
- Email

View the discussion thread.