ICEcoder / classes / **Settings.php** / Jump to

mattpass Dynamic assets path, tweak and fix in settings, ignore .idea **History**

1 contributor

245 lines (221 sloc) | 8.43 KB

```php
<?php declare(strict_types=1);

namespace ICEcoder;

class Settings
{
    public function __construct()
    {
        // Set version number and document root as core settings
        // Defaults to the right
        $this->versionNo = "8.1";                           // "8.1";
        $this->docRoot = $_SERVER['DOCUMENT_ROOT'];         // $_SERVER['DOCUMENT_ROOT']
        $this->assetsRoot = "assets";                       // "assets" (relative or absolute)
    }

    public function getCoreDetails()
    {
        return [
            "versionNo" => $this->versionNo,
            "docRoot" => $this->docRoot,
            "assetsRoot" => $this->assetsRoot,
        ];
    }

    // ========
    // DATA DIR
    // ========

    public function getDataDirDetails()
    {
        clearstatcache();

        // Return details about the data dir
        $fullPath = dirname(__FILE__) . "/../data/";
        $exists = file_exists($fullPath);
        $readable = is_readable($fullPath);
        $writable = is_writable($fullPath);
        return [
            "fullPath" => $fullPath,
            "exists" => $exists,
            "readable" => $readable,
            "writable" => $writable,
        ];
    }

    // =============
    // GLOBAL CONFIG
    // =============

    public function getConfigGlobalTemplate($asArray)
    {
        // Return the serialized global config template
        $fileName = 'template-config-global.php';
        $fullPath = dirname(__FILE__) . "/../lib/" . $fileName;
        if (function_exists('opcache_invalidate')) {
            opcache_invalidate($fullPath, true);
        }
        $settings = file_get_contents($fullPath);
        if ($asArray) {
            $settings = $this->serializedFileData("get", $fullPath);
        }
        return $settings;
    }

    public function getConfigGlobalFileDetails()
    {
        clearstatcache();

        // Return details about the global config file
        $fileName = 'config-global.php';
        $fullPath = dirname(__FILE__) . "/../data/" . $fileName;
        $exists = file_exists($fullPath);
        $readable = is_readable($fullPath);
        $writable = is_writable($fullPath);
        $filemtime = filemtime($fullPath);
        return [
            "fileName" => $fileName,
            "fullPath" => $fullPath,
```

```php
            "exists" => $exists,
            "readable" => $readable,
            "writable" => $writable,
            "filemtime" => $filemtime
        ];
    }

    public function getConfigGlobalSettings()
    {
        // Start an array with version number and document root
        $settings = $this->getCoreDetails();
        // Get global config file details
        $fullPath = $this->getConfigGlobalFileDetails()['fullPath'];
        $settingsFromFile = $this->serializedFileData("get", $fullPath);
        // Merge that with the array we started with and return
        $settings = array_merge($settings, $settingsFromFile);
        return $settings;
    }

    public function setConfigGlobalSettings($settings): bool
    {
        // Get the global config file details
        $fullPath = $this->getConfigGlobalFileDetails()['fullPath'];
        if ($fConfigSettings = fopen($fullPath, 'w')) {
            // If the settings we've received aren't in serialized format yet, do that now
            // As $settings could be a serialized string or array
            if (is_array($settings)) {
                unset($settings['versionNo']);
                unset($settings['docRoot']);
            }
            return $this->serializedFileData("set", $fullPath, $settings);
        } else {
            return false;
        }
    }

    public function updateConfigGlobalSettings($array): bool
    {
        // Update global config settings file
        $settingsFromFile = $this->getConfigGlobalSettings();
        $settings = array_merge($settingsFromFile, $array);
        return $this->setConfigGlobalSettings($settings);
    }

    // ============
    // USERS CONFIG
    // ============

    public function getConfigUsersTemplate($asArray)
    {
        // Return the serialized users config template
        $fileName = 'template-config-users.php';
        $fullPath = dirname(__FILE__) . "/../lib/" . $fileName;
        if (function_exists('opcache_invalidate')) {
            opcache_invalidate($fullPath, true);
        }
        $settings = file_get_contents($fullPath);
        if ($asArray) {
            $settings = $this->serializedFileData("get", $fullPath);
        }
        return $settings;
    }

    public function getConfigUsersFileDetails($fileName)
    {
        // Return details about the users config file
        $fullPath = dirname(__FILE__) . "/../data/" . $fileName;
        $exists = file_exists($fullPath);
        $readable = is_readable($fullPath);
        $writable = is_writable($fullPath);
        $filemtime = filemtime($fullPath);
        return [
            "fileName" => $fileName,
            "fullPath" => $fullPath,
            "exists" => $exists,
            "readable" => $readable,
            "writable" => $writable,
            "filemtime" => $filemtime,
        ];
    }

    public function getConfigUsersSettings($fileName)
    {
        // Get users config file details
        $fullPath = $this->getConfigUsersFileDetails($fileName)['fullPath'];
        $settingsFromFile = $this->serializedFileData("get", $fullPath);
        // Now return
        return $settingsFromFile;
    }

    public function setConfigUsersSettings($fileName, $settings): bool
    {
        // Get the users config file details
        $fullPath = $this->getConfigUsersFileDetails($fileName)['fullPath'];
        if ($fConfigSettings = fopen($fullPath, 'w')) {
            return $this->serializedFileData("set", $fullPath, $settings);
        } else {
            return false;
        }
    }

    public function updateConfigUsersSettings($fileName, $array): bool
    {
        // Update users config settings file
        $settingsFromFile = $this->getConfigUsersSettings($fileName);
        $settings = array_merge($settingsFromFile, $array);
        return $this->setConfigUsersSettings($fileName, $settings);
    }

    public function updateConfigUsersCreateDate($fileName)
    {
        global $ICEcoderUserSettings;

        // Get users config file details
        $filemtime = $this->getConfigUsersFileDetails($fileName)['filemtime'];
        // Make it a number (avoids null, undefined etc)
        $filemtime = intval($filemtime);
        // Set it to the epoch time now if we don't have a real value
        if (0 === $filemtime) {
            $filemtime = time();
        }
        // Update users config settings file
        $ICEcoderSettingsFromFile = $this->getConfigUsersSettings($fileName);
        $ICEcoderSettingsFromFile['configCreateDate'] = $filemtime;
        $this->setConfigUsersSettings($fileName, $ICEcoderSettingsFromFile);
        // Set the new value in array
        $ICEcoderUserSettings['configCreateDate'] = $filemtime;
    }

    public function createIPSettingsFileIfNotExist()
    {
        global $username, $settingsFile, $settingsFileAddr;

        // Create a duplicate version for the IP address of the domain if it doesn't exist yet
        $serverAddr = $_SERVER['SERVER_ADDR'] ?? "1";
        if ($serverAddr == "1" || $serverAddr == "::1") {
            $serverAddr = "127.0.0.1";
        }
        $settingsFileAddr = 'config-' . $username . str_replace(".", "_", $serverAddr) . '.php';
        if (false === file_exists(dirname(__FILE__) . "/../data/" . $settingsFileAddr)) {
            if (false === copy(dirname(__FILE__) . "/../data/" . $settingsFile, dirname(__FILE__)
                $reqsFailures = ["phpCreateSettingsFileAddr"];
                include dirname(__FILE__) . "/../lib/requirements.php";
            }
        }
    }

```

```php
    public function serializedFileData($do, $fullPath, $output=null)
    {
        if ("get" === $do) {
            if (function_exists('opcache_invalidate')) {
                opcache_invalidate($fullPath, true);
            }
            $data = file_get_contents($fullPath);
            $data = str_replace("<"."?php\n/*\n\n", "", $data);
            $data = str_replace("\n\n*/\n?".">", "", $data);
            $data = unserialize($data);
            return $data;
        }
        if ("set" === $do) {
            if (true === is_array($output)) {
                $output = serialize($output);
            }
            return false !== file_put_contents($fullPath, "<"."?php\n/*\n\n" . $output . "\n\n*/\n
        }
    }
}
```