dump_project.php ?????????????? ????????? ??????

Запрос :

```
для архивирования проектов я использую dump_project.php
мне нужно скопировать этот скрипт в корень проекта и сделать при его помощи дамп.
Проектов много и трудно следить за актуальностью этого скрипта.

Как можно сделать так. чтобы скрипт был в одном месте, а подгружался в проекты автоматически.
Можно ли для этого использовать composer ?
Можно ли как то "прикрутить" github  , чтобы с него тянуть последнюю версию?
Или какие вообще есть варианты?
```

Ответ ИИ Отличная задача! Действительно, неудобно хранить один и тот же скрипт во многих проектах. Вот несколько проверенных решений:

1. ***????? Composer (?????????????? ??????)***

Создайте отдельный composer-пакет для вашего скрипта.

????????? ??????:

```
your-dump-tool/
├── composer.json
├── bin/
│   └── dump-project      # исполняемый скрипт
└── src/
    └── Dumper.php
```

`composer.json`:

```
{
    "name": "your-vendor/dump-tool",
    "type": "composer-plugin", // или просто "library"
    "bin": ["bin/dump-project"],
    "require": {
        "php": ">=7.4"
    },
    "autoload": {
        "psr-4": {
            "YourVendor\\DumpTool\\": "src/"
        }
    }
}
```

????????? ? ????????:

```
composer require --dev your-vendor/dump-tool
```

После этого скрипт будет доступен в `vendor/bin/dump-project`

??????:

```
php vendor/bin/dump-project
# или если добавить в composer.json проекта:
composer dump-project
```

2. ***????? GitHub + ?????????????***

Создайте репозиторий на GitHub, а в проектах используйте **скрипт-загрузчик**:

`load_dumper.php` (*??????? ? ?????? ??????? ???????*):

```php
<?php
$dumperUrl = 'https://raw.githubusercontent.com/yourname/php-dumper/main/dump_project.php';
$localPath = __DIR__ . '/dump_project_temp.php';

// Загружаем последнюю версию
$content = file_get_contents($dumperUrl);
if ($content === false) {
    die("Не удалось загрузить дампер\n");
}

// Проверяем хэш для кэширования (опционально)
$cacheFile = __DIR__ . '/.dumper_cache';
if (file_exists($cacheFile) && md5($content) === file_get_contents($cacheFile)) {
    // Используем существующий
    require_once __DIR__ . '/dump_project.php';
    exit;
}

// Сохраняем и запускаем
file_put_contents($localPath, $content);
file_put_contents($cacheFile, md5($content));

require_once $localPath;
// Не забываем удалить после выполнения
unlink($localPath);
```

3. ?????????? composer-?????

Установите скрипт глобально на сервере:

```
composer global require your-vendor/dump-tool
```

А в каждом проекте создайте символическую ссылку:

```
ln -s ~/.composer/vendor/bin/dump-project /path/to/project/dump_project.php
```

Или добавьте глобальный bin в PATH и вызывайте откуда угодно:

```
dump-project /path/to/project
```

4. **Bash-??????-??????? (??????? ???????)**

Создайте скрипт на сервере:

`/usr/local/bin/dump-project`:

```
#!/bin/bash
DUMPER_REPO="https://raw.githubusercontent.com/yourname/php-dumper/main/dump_project.php"
TEMP_DUMPER="/tmp/dump_project_$$.php"

# Скачиваем последнюю версию
curl -s "$DUMPER_REPO" > "$TEMP_DUMPER"

# Запускаем в контексте целевого проекта
cd "$1" || exit 1
php "$TEMP_DUMPER"

# Чистим
rm "$TEMP_DUMPER"
```

Использование:

```
dump-project /path/to/project
```

5. **Git submodule (???? ?????? ????? ????????)**

```
cd /path/to/project
git submodule add https://github.com/yourname/dump-tool.git tools/dump-tool
ln -s tools/dump-tool/dump_project.php dump_project.php
```

Для обновления всех проектов:

```
git submodule update --remote
```

??????????:

1. **Для командной разработки → Composer** (проще всего управлять версиями)
2. **Для личного использования → Глобальный composer-пакет + симлинки**
3. **Если нет доступа к composer → Bash-скрипт с curl**

????????? ??????? ? ????????????:

```
// dump_runner.php
<?php
class DumpRunner {
    private $repoUrl = 'https://api.github.com/repos/yourname/dump-
tool/contents/dump_project.php';
    private $cacheFile;
    private $localScript;

    public function __construct($projectRoot) {
        $this->cacheFile = $projectRoot . '/.dumper_meta.json';
        $this->localScript = $projectRoot . '/dump_project.php';
    }

    public function run() {
        if ($this->needUpdate()) {
            $this->downloadLatest();
        }
        require_once $this->localScript;
    }

    private function needUpdate() {
        if (!file_exists($this->localScript)) return true;

        // Проверяем SHA файла через GitHub API
        $ch = curl_init("$this->repoUrl?ref=main");
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
```

```
        curl_setopt($ch, CURLOPT_USERAGENT, 'DumpRunner');
        $response = json_decode(curl_exec($ch), true);

        $currentSha = file_exists($this->cacheFile) ?
                      json_decode(file_get_contents($this->cacheFile))->sha : null;

        return $currentSha !== $response['sha'];
    }
}

$runner = new DumpRunner(__DIR__);
$runner->run();
```

Итог: Лично я рекомендую **Composer + bin** — это стандарт в PHP-мире, легко обновляется через `composer update`, и все разработчики поймут этот подход.

Revision #4
Created 23 April 2026 15:19:11 by mcsweb
Updated 23 April 2026 15:24:07 by mcsweb