Composer Pakete für TER Upload in Extensions integrieren

Warum einfach, wenn’s auch kompliziert geht?

TYPO3-Slack @alexander.grein | alexander.grein@gmail.com

Warum Composer Pakete nutzen?

Warum?

Welche Vorteile bietet die Nutzung von Composer Paketen?

- Das Rad nicht neu erfinden
- Einfache Nutzung von fertigen Bibliotheken
- Einfache Aktualisierung von Bibliotheken
- Automatische Auflösung von Abhängigkeiten
- Autoloading
- …

Was sind die Nachteile?

Nachteile?

Was sind die Nachteile der Nutzung von Composer Paketen?

- Extension lässt sich nur über Composer installieren
- Extension erscheint dadurch nicht im TER, da dort (im Moment) nur Extensions veröffentlich werden können, die auch die „klassische“ Installation über den Extension-Manager unterstützen

Was tun?

Was tun?

Grundidee

- Die Composer Pakete müssen in die Extension integriert werden, aber nur in die TER-Version
- Die integrierten Pakete müssen von TYPO3 geladen werden, aber nur in Nicht-Composer-Installationen

Wie?

Wie?

Wie integriert man Composer Pakete in eine Extension?

- Man erzeugt eine PHAR-Datei („PHP-Zip“)
- Man integriert die erzeugte PHAR-Datei in die TER-Version der Extension
- Man lädt die in der PHAR-Datei enthaltene autoload.php in der eigenen Extension, sofern sie in einer Nicht-Composer-TYPO3-Umgebung installiert ist

Step by Step

Werkzeugkasten

Wie erzeugt man PHAR-Dateien?

Möglichkeit 1

humbug/box

https://github.com/box-project/box

Installation

https://github.com/box-project/box/blob/main/doc/installation.md

Werkzeugkasten

Wie erzeugt man PHAR-Dateien?

Möglichkeit 2

clue/phar-composer

https://github.com/clue/phar-composer

Installation

Konsole

```
> composer global require clue/phar-composer
```

Hinweis:
$HOME/.composer/vendor/bin in $PATH setzen!

Configuration

Configuration

Benötigte Dateien anlegen

Beispiel:

```
Resources/Private/PHP/.gitignore
Resources/Private/PHP/composer.json
```

Nur für humbug/box benötigt:

```
Resources/Private/PHP/box.json
```

Configuration

.gitignore

Beispiel:

```
*.phar
vendor
```

Configuration

composer.json

Beinhaltet alle abhängigen Pakete, die nicht bereits von TYPO3 mitgeliefert werden.
Siehe vendor-Verzeichnis von Nicht-Composer-TYPO3-Installation.

Beispiel:

```
{
  "description": "Dependencies",
  "config": {
    "optimize-autoloader": true,
    "prepend-autoloader": false,
    "classmap-authoritative": true
  },
  "require": {
    "tedivm/fetch": "^0.7",
    "league/html-to-markdown": "^5"
  }
}
```

Configuration humbug/box

box.json

```
{
  "blacklist": [
    "box.json",
    "vendor.phar"
  ],
  "main": false,
  "output": "vendor.phar"
}
```

PHAR-Datei erzeugen

humbug/box

Pakete installieren

```
composer install --no-dev --working-dir=Resources/Private/PHP
```

Pakete updaten

```
composer update --no-dev --working-dir=Resources/Private/PHP
```

PHAR-Datei erzeugen

```
box compile -c Resources/Private/PHP/box.json
```

PHAR-Datei erzeugen

clue/phar-composer

Pakete installieren

```
composer install --no-dev --working-dir=Resources/Private/PHP
```

Pakete updaten

```
composer update --no-dev --working-dir=Resources/Private/PHP
```

PHAR-Datei erzeugen

```
phar-composer build Resources/Private/PHP/composer.json Resources/Private/PHP/vendor.phar
```

PHAR-Datei für einzelnes Paket anlegen

```
phar-composer build league/html-to-markdown Resources/Private/PHP/html-to-markdown.phar
```

Problem:
Es werden auch eventuell vorhandene dev-Dependencies der Pakete mit eingebunden

Makefile

Makefile

humbug/box

```
.PHONY: update-build-vendor-phar
update-build-vendor-phar:
	composer update --no-dev --working-dir=Resources/Private/PHP
	box compile -c Resources/Private/PHP/box.json

.PHONY: build-vendor-phar
build-vendor-phar:
	composer install --no-dev --working-dir=Resources/Private/PHP
	box compile -c Resources/Private/PHP/box.json
```

Makefile

clue/phar-composer

```
.PHONY: update-build-vendor-phar
update-build-vendor-phar:
    composer update --no-dev --working-dir=Resources/Private/PHP
    phar-composer build Resources/Private/PHP/composer.json Resources/Private/PHP/vendor.phar
.PHONY: build-vendor-phar
build-vendor-phar:
    composer install --no-dev --working-dir=Resources/Private/PHP
    phar-composer build Resources/Private/PHP/composer.json Resources/Private/PHP/vendor.phar
```

PHAR Integration

PHAR-Integration

Einbindung des erzeugen PHAR-Files

Über ext_localconf.php

```
if (!\TYPO3\CMS\Core\Core\Environment::isComposerMode()) {
   $vendorPharFile = \TYPO3\CMS\Core\Utility\GeneralUtility::getFileAbsFileName('EXT:extensionkey/Resources/Private/PHP/vendor.phar');
   if (file_exists($vendorPharFile)) {
      require 'phar://' . $vendorPharFile . '/vendor/autoload.php';
   }
}
```

Innerhalb einer PHP-Klasse, die eine Abhängigkeit (z.B. Fetch\Message) besitzt

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

namespace VENDOR\Extensionkey\Command;

use Fetch\Message;
use TYPO3\CMS\Core\Core\Environment;
use TYPO3\CMS\Core\Utility\ExtensionManagementUtility;

if (!Environment::isComposerMode() && !class_exists(Message::class)) {
   require_once 'phar://' . ExtensionManagementUtility::extPath('extensionkey') . 'Resources/Private/PHP/vendor.phar/vendor/autoload.php';
}
```

Neue TER Version erzeugen

Neue TER Version erzeugen

Step by Step

1. Neue Versionsnummer (z.B. 1.0.0) in die ext_emconf.php der Extension eintragen (ohne Präfix "v")
2. Alle Änderungen commiten, **ohne** PHAR-Datei und dem dafür erzeugten vendor-Ordner! (siehe .gitignore-Dateien)
3. Git-Tag mit gleicher Versionsnummer, aber mit Präfix "v" erstellen (z.B. v1.0.0) und Änderungen inkl. Tag ins Git-Repository der Extension pushen
4. Zip-Datei der Extension erzeugen, welche die erzeugte PHAR-Datei beinhaltet
5. Die erzeugte Datei im Format extensionkey_1.0.0.zip ins TER hochladen

Makefile

Extension ZIP erzeugen

```
.PHONY: zip
zip:
    grep -Po "(?<='version' => ')([0-9]+\.[0-9]+\.[0-9]+)" ext_emconf.php | xargs -I {version} \
    sh -c 'rm ../zip/$(shell basename $(CURDIR))_{version}.zip || true; \
    mkdir  -p ../zip; git archive -v -o "../zip/$(shell basename $(CURDIR))_{version}.zip" v{version};'
    grep -Po "(?<='version' => ')([0-9]+\.[0-9]+\.[0-9]+)" ext_emconf.php | xargs -I {version} \
    sh -c 'zip "../zip/$(shell basename $(CURDIR))_{version}.zip" \
    Resources/Private/PHP/vendor.phar'
```

Hinweis für macOS Nutzer:
Der dort verfügbare grep-Befehl kennt den Parameter -P nicht.
Damit das obige Script funktioniert, muss der grep-Befehl daher über brew installiert werden:
> brew install grep
Damit der System-Befehl „grep“ so wie bisher funktioniert, hat der über den brew installer installierte grep den Namen „ggrep“.
Entsprechend muss der grep-Befehl in dem Makefile durch ggrep ersetzt werden.

Git-Workflow

Beispiel EXT:typo3-warming

https://github.com/eliashaeussler/typo3-warming/blob/0.4.8/.github/workflows/release.yaml

Herzlichen Dank

für die Pionierarbeit und Unterstützung!

Helmut Hummel

Chris Müller

Elias Häußler

Blog-Artikel zum Thema von Helmut Hummel:
https://insight.helhum.io/post/148112375750/how-to-include-binaries-for-ter-exts

Fragen?

Danke!