Fileuploader

Abnahme des IHK-Projekts von Fabian Michael

Stand: 03 November 2020

Inhalt

1 Was wurde getestet ... 3
1.1 IE-11 Kombatibilität ... 3
1.1.1 Fehler in der Konsole ... 3
1.1.2 Nicht testbar im IE-11 ... 3
1.2 FE-Test im Cromium(V 80.0.)/Firefox(V 82.0.) ... 3
1.3 BE-Test ... 3
1.3.1 BE-Validierung von Mime-Types funktioniert nicht. ... 3
1.3.2 BE Validierung von der Max-Filesize funktioniert nicht ... 3
1.3.3 BE Validierung Max-Files funktioniert ... 4
1.3.4 Hochladen der Dateien auf den Server ... 4
2 Code Review ... 5
2.1 Allgemeine Code-Anmerkungen ... 5
2.2 ext_localconf.php (form4_fileuploader) ... 5
2.3 ViewHelper ... 5
2.3.1 Fehler/Offene Punkte ... 5
2.3.2 Positive Anmerkungen ... 5
2.3.3 Offene ToDo's ... 5
2.4 Review zu unterschiedlichen Processor-Klassen ... 5
2.4.1 IFIleuploadProcessor ... 5
2.4.2 AbstactFileuploadProcessor ... 5
2.4.3 FileuploadDestinationProcessor ... 5
2.4.4 FileuploadFileExtensionProcessor ... 5
2.4.5 FileuploadMaxFileSizeProcessor ... 6
2.4.6 FileuploadMaxFilesProcessor ... 6
2.4.7 FileuploadMaxUploadSizeProcessor ... 6
2.4.8 FileuploadMIMETypeProcessor ... 6
2.5 Configuration-Api ... 6
2.5.1 Configuration ... 6
2.5.2 ConfigurationRegistry ... 7
2.6 FileuploadService ... 7
3 Probleme der aktuellen Umsetzung ... 8
3.1 Anforderungen ... 8
3.2 Aktueller Stand ... 8
3.3 Vorschlag zur Erweiterung des aktuellen Fileuploaders ... 8
3.3.1 Entkoppelung des ViewHelpers ... 8
3.3.2 Konfiguration der FE-Validierung ... 8
4 Fazit zur Abnahme ... 9
4.1 Probleme, welche auf jeden Fall behoben werden müssen ... 9
4.2 Fehler, mit denen die Abnahme unter Vorbehalt erfolgen kann ... 9

1 Was wurde getestet

1.1 IE-11 Kombatibilität

Im IE-11lief der Fileuploader nicht

1.1.1 Fehler in der Konsole

Im IE-11 werden Fehler in der Konsole geworfen

- „globalThis is undefined“ in der form4-fileuploader.js
- es gibt einen Syntax-Fehler in der test.js

1.1.2 Nicht testbar im IE-11

- Keine Preloaded-Files sichtbar
- Beim Submit passiert nichts, es kommt ein leerer String zurück

1.2 FE-Test im Cromium(V 80.0.)/Firefox(V 82.0.)

- Max+Min-Files funktioniert
  - Preloaded-Files werden mitgezählt
  - Wenn Files gelöscht werden, wird der Zähler aktualisiert
- Max-Filesize funktioniert
- Min-Filesize nicht konfiguriert
- Max-Upload-Size/Fileextension sind im FE nicht konfiguriert
- Mime-Type ist im FE konfiguriert.
  - Funktioniert aktuell nicht
  - Worddatei mit Endung .pdf wird akzeptiert
  - Null-Byte Datei wird akzeptiert
  - Leere Dateien werden akzeptiert

1.3 BE-Test

1.3.1 BE-Validierung von Mime-Types funktioniert nicht.

- Mime-Type ist im FE konfiguriert (Deswegen gehen wir davon aus, dass diese auch im BE ausgeführt wird). Allerdings gibt es im BE keine Mime-Type Validierung
  - Wird im BE
  - Worddatei mit Endung .pdf wird akzeptiert.
  - Null-Byte Datei wird akzeptiert.
  - Leere Dateien werden akzeptiert.

1.3.2 BE Validierung von der Max-Filesize funktioniert nicht

Es können Dateien beliebiger Größe hochgeladen werden, wenn die FE-Validierung ausgeschaltet ist.

1.3.3 BE Validierung Max-Files funktioniert

Getestet, Alles ok

1.3.4 Hochladen der Dateien auf den Server

- Das Hochladen der Dateien funktioniert

2 Code Review

2.1 Allgemeine Code-Anmerkungen

- PSR Coding-Guidlines sollten eingehalten werden, z.B.
  - zwischen if und (gehört ein Leerzeichen
  - Keine einzeiligen If-Blöcke
  - Keine einzeiligen PhpDoc-Blöcke für Methoden, Properties usw. (Ausnahme sind Variablen innerhalb von Codeblöcken)
- Private Methoden sollte es nur mit gutem Grund geben. Protected ist vorzuziehen.

2.2 ext_localconf.php (form4_fileuploader)

- Hier sollte auf jeden Fall ein Beispiel für eine Konfiguration dokumentiert werden

2.3 ViewHelper

2.3.1 Fehler/Offene Punkte

- Exception im View-Helper: bei einer Fehl-Konfiguration sollte das Seitenrendering nicht unterbrochen werden
- PSR-Richtlinien OK, aber nicht 100%Positive Anmerkungen
- Es werden viele Array-Methoden verwendet

2.3.2 Offene ToDo's

- Prüfen, ob die Implementierung mit render() den best-Practices entspricht. Evtl. sollte renderStatic() verwendet werden.

2.4 Review zu unterschiedlichen Processor-Klassen

2.4.1 IFIleuploadProcessor

- PHP-Docs generieren. Gerade in Interfaces ist eine gute Dokumentation entscheidend.

2.4.2 AbstactFileuploadProcessor

Gut

2.4.3 FileuploadDestinationProcessor

Gut

2.4.4 FileuploadFileExtensionProcessor

- Z.27-29 werden mehrfach verwendet. Hier könnte man eine kleine Utility mit einer static function schreiben.
- Z.27 - Es gibt mit Sicherheit eine Möglichkeit, einen String als Regex auszuführen und gleichzeitig herauszufinden, ob es sich dabei eine valide Regex handelt. D.h. preg_match muss nur einmal ausgeführt werden.
- Konnte bisher nicht auf Funktionalität überprüft werden, da nicht konfiguriert

2.4.5 FileuploadMaxFileSizeProcessor

- funktioniert aktuell nicht

2.4.6 FileuploadMaxFilesProcessor

Gut

2.4.7 FileuploadMaxUploadSizeProcessor

- Konnte bisher nicht getestet werden
- Da der FileuploadMaxFileSizeProcessor nicht funktioniert, gehe ich davon aus dass dieser Processor auch nicht funktional ist, da hier die gleiche Funktion verwendet wird.

2.4.8 FileuploadMIMETypeProcessor

- Funktioniert aktuell nicht
- Z28-31 siehe auch FileuploadFileExtensionProcessor: in eine Utility auslagern
- Wie sinnvoll ist es den Mime-Type im FE zu testen.
  - Ist dieser im FE überhaupt bekannt?
  - Sollte im FE nicht stattdessen die File-Extension überprüft werden?

2.5 Configuration-Api

2.5.1 Configuration

- Aktuell wird in der addSettings-Methode der $settingsKey benötigt,
  - dieser hat Auswirkungen auf den View-Helper und das FE.
  - Wenn diese Abhängigkeit aufgelöst ist, wird dieses Argument nicht mehr gebraucht.

Aussicht

Die Configuration-Klasse sollte etwas refactored werden:

- Statt immer das gesamte Configuration-Array zu übergeben, sollten die Processor's einzeln geadded werden
- Es sollte statt der $backendOnly Setting zwei Settings verwendet werden (Falls noch benötigt)
  - $enableBackend
  - $enableFrontend
- Die Configuration sollte eine Methode wie folgt implementiert werden:

```
Public function addProcessor(string $class, array $settings = [], bool
$enableBackend = true, bool $enableFrontend = true) : Configuration;
```

- Diese Methode sollte auch in der ext_localconf.php für jeden Processor aufgerufen werden

Wir wünschen uns folgende API:

```
$registry->registerConfiguration('test')
    ->addProcessor(AProcessor::class, ['something' => 'value'])
    ->addProcessor(AnotherProcessor::class, [], true, false);
```

2.5.2 ConfigurationRegistry

Gut

2.6 FileuploadService

- Für Exception wird kein use benötigt, es kann als \Exception verwendet werden.
- Z 51: Variablen $settingsKey und $settings sollten besser benannt werden
- Die function validateChecksum sollte nicht public sein
- Die Exception, welche in der Function calculateChecksum geworfen wird, wird aktuell nicht behandelt
- handleRemove bietet keine Schnittstelle zu den Processoren
  - Falls Files von dem Server entfernt werden sollen, muss der Service angepasst werden. Hier wäre eine Schnittstelle besser.
  - Muss nicht in dieser Iteration implementiert werden. Auf jeden Fall sollte aber ein TODO erstellt werden.
- Hast du überprüft, ob sha1 weiterhin verwendet werden sollte, oder ob wir auf einen besseren Algo umsteigen sollten.

3 Probleme der aktuellen Umsetzung

3.1 Anforderungen

Folgende Anforderungen wurden gestellt

- Gefordert war ein Single Point of Configuration
- Gefordert war, dass die Konfiguration beliebig durch einen Entwickler erstellt werden kann

3.2 Aktueller Stand

Leider wurde die Vorgabe des Single Points of Configuration durch den Kunden unterschätzt. Hier gab es auch ungenügende Vorgaben.

- Aktuell muss die BE und die FE Konfiguration einzeln erstellt werden
- Es gibt Konfiguration, die gar nicht angepasst werden können, da diese hart Kodiert sind.

3.3 Vorschlag zur Erweiterung des aktuellen Fileuploaders

3.3.1 Entkoppelung des ViewHelpers

Problem

- Aktuell sind min_files und max_files als Konfigurations-Keys hart kodiert
- In der ext_localconf.php wird jedoch suggeriert, dass diese frei wählbar sind

Lösungsvorschlag

- Die einzelnen Prozessoren müssen den ViewHelper hooken können
- Dazu muss das IFileuploadProcessor + die implementierenden Klassen erweitert werden

3.3.2 Konfiguration der FE-Validierung

Problem:

- Aktuell wird die FE-Validierung im JS einzeln konfiguriert
- Die Validierung wird durch die Processor-Keys gesteuert
- Wählt ein Entwickler andere Keys, so wird die FE-Validierung nicht ausgeführt

Lösungsvorschlag

- Die einzelnen Processoren sollten eine Funktion implementieren, die die notwendigen JS-Konfigurationen zurückgibt
- Dazu muss das IFileuploadProcessor + die implementierenden Klassen erweitert werden
- Das JS muss angepasst werden.

4 Fazit zur Abnahme

4.1 Probleme, welche auf jeden Fall behoben werden müssen

- Die Fehler aus Punkt 2 + 3 sollten auf jeden Fall vorher behoben werden.
- Ansonsten kann eine Abnahme nicht erfolgen, da die Grundfunktion nicht sichergestellt ist.

4.2 Fehler, mit denen die Abnahme unter Vorbehalt erfolgen kann

- Die Fehler im Punkt 4 müssen auf jeden Fall in der nächsten Version des File-Uploaders behoben werden
- Das Feature soll jedoch möglichst bald im Produktivbetrieb eingesetzt werden
- Wir schätzen den Aufwand, der bei den gewünschten Änderungen auf unserer Seite durch die Änderung der API entsteht, als gering ein:
  - Es müssen Konfigurationsdateien angepasst werden
  - Eine Migrations-Anleitung muss seitens Entwickler erstellt werden
- Somit kann eine Abnahme einer ersten Version der Uploaders unter Vorbehalt erfolgen, obwohl die Fehler aus Punkt 4 noch nicht behoben sind.