1 Paketdefinitionen

Sie können Komponenten und andere Dateien zu Paketen zusammenstellen, indem Sie eine entsprechende Definition im /Plugins Ordner erstellen (im json Format). Der Installationsassistent wird diese selbstständig erkennen und verwenden.

1.1 Aufbau

Die Definition erfolgt im json Format.

- name Der Name des Pakets wird im Installationsassistenten angezeigt
- version Die Versionsnummer wird für die Überprüfung der requirements benötigt. Dabei sollten die einzelnen Versionssegmente durch einen . (Punkt) getrennt werden.
- versionDate Ein zur Versionsnummer zugehöriges Datum !!!keine Verwendung!!!
- author Der Verfasser des Pakets/ der Komponenten (eventuell eine Aufzählung) !!!keine Verwendung!!!
- sourceURL Die Quelladresse des Pakets !!!keine Verwendung!!!
- updateURL Diese Adresse sollte Aktualisierungsinformationen bereitstellen !!!keine Verwendung!!!
- requirements Die Voraussetzungen sollen Mindestanforderungen an das übrige System beschreiben. Dabei können derzeit lediglich andere Pakete als Voraussetzung bestimmt werden.
  - type Als Typ ist lediglich plugin möglich
  - name Der Name des entsprechenden Komponente/Paket, welches die Voraussetzung bilden soll
  - version eine eventuelle Versionsnummer der Komponente oder des Pakets, welche mindestens erfüllt sein muss.
- files Hier können einzelne Dateien und Ordner, vom Grundverzeichnis ausgehend,aufgezählt werden. Dabei sollte auch die Definitionsdatei des Paketes enthalten sein. Dateien können durch Pfad/Dateiname.Endung und Ordner mittels Pfad/Ordner/ dem Paket hinzugefügt werden.

- components Hier können Komponentendefinitionen aufgelistet werden, deren Inhalt verwendet werden soll.
  - conf Dieses Feld gibt die Adresse der Definition, vom Grundverzeichnis aus, an.

1.2 Beispiel

ostepu-db.json

```
{
  "name":"OSTEPU-DB",
  "version":"0.3",
  "versionDate":"19.04.2015",
  "author":"",
  "sourceURL":"",
  "updateURL":"",
  "requirements":[{"type":"plugin","name":"CORE","version":"0.2"
     }],
  "files":
    [
      {"path":"Plugins/ostepu-db.json"},
      {"path":"Assistants/"},
      {"path":"DB/.htaccess"}
    ],
  "components":
    [
      {"conf":"DB/DBCourse/Component.json"},
      {"conf":"DB/DBCourseStatus/Component.json"},
      {"conf":"DB/DBExercise/Component.json"}
    ]
}
```

2 Komponentendefinitionen

Der Aufbau einer Komponente wird in der zugehörigen Components.json beschrieben.

2.1 Aufbau

- name Der Name der Komponente (dieser muss eindeutig sein)
- version Die Versionsnummer wird für die Überprüfung der requirements von Paketen (eventuell auch Komponente) benötigt. Dabei sollten die einzelnen Versionssegmente durch einen . (Punkt) getrennt werden. !!!keine Verwendung!!!
- classFile Sofern die Komponente auch als Objekt aufrufbar sein soll, muss der Name der Klassendatei und der Name der aufzurufenden Klasse angegeben werden.
- className Der Name der Hauptklasse, die Erzeugung einer Instanz der Klasse soll die Komponente auslösen
- files Hier können einzelne Dateien und Ordner, vom Grundverzeichnis ausgehend,aufgezählt werden. Dateien können durch Pfad/Dateiname.Endung und Ordner mittels Pfad/Ordner/ der Komponente hinzugefügt werden.
- type Sofern es sich nicht um eine normale Definition handelt, kann hier der Typ clone zum verwenden einer Komponentenkopie angegeben werden.
- base Wenn es sich bei dieser Komponente um eine Kopier einer anderen Komponente handelt, muss die kopierte Komponente angegeben werden (Name), sodass deren Definition bei der Installation geladen werden kann.
- baseURI Für Kopien von Komponente (wenn diese eine ist), muss einer eindeutiger Aufrufpräfix angegeben werden.
- option Hier können eventuelle Einstellungsmöglichkeiten der Komponenten durch , (Komma) getrennt aufgelistet werden.
- links Hier werden Verbindungen von dieser Komponente, zu anderen Komponenten, beschrieben.
  - name Der Name des Ausgangs. Dieser Name wird von anderen Komponenten verwendet, wenn sich diese mit diesem Ausgang verbinden wollen.
  - target Hier kann eine einzelne Komponente oder eine Liste von Bezeichner genannt werden, zu denen sich diese Komponente an diesem Ausgang verbinden möchte. Die Aufzählung kann aber auch leer bleiben, sofern der Ausgang nur für das Anknüpfen anderer bereitgestellt werden soll.
  - links In diesem Bereich werden mögliche Aufrufe festgelegt, welche über

diesen Ausgang abgewickelt werden können sollen. Sodass der Installationsassistent prüfen kann, ob eventuelle Zielkomponenten diese anbieten.

- connector Hier werden Verbindungen beschrieben, welche diese Komponente von anderen Komponenten zu sich selbst fordert.
  - name Der Name des Ausgangs der Komponente
  - target Der Name der Komponente, welche über den in name genannten Ausgang diese Komponente aufrufen soll

2.2 Beispiel

```
{
    "name": "SampleA",
    "version": "1.0",
    "classFile": "SampleA.php",
    "className": "SampleA",
    "files": [
        {"path": ".htaccess"},
        {"path": "SampleA.php"},
        {"path": "index.php"},
        {"path": "Commands.json"}
    ],
    "links": [
        {
            "name": "outA",
            "target": "SampleB",
            "links": [
                {
                    "method": "GET",
                    "path": "/callMe/:abc"
                }
            ]
        }
    ],
    "connector": [
        {
            "name": "outB",
            "target": "SampleC"
        }
    ]
}
```

Abbildung 2.1: SampleC ist über den Ausgang outB mit SampleA verbunden und SampleA über outA mit SampleB

2.3 Kopien

Es gibt Komponenten, welche die Definition weiterer Komponenten durch das hinzufügen eines eindeutigen Präfixes erlauben. Dabei kann die Kopie mitunter eigene Tabellen (auf Datenbanken bezogen) oder Pfade verwenden. Beispiele für solche kopierbare Komponenten sind CAbstract und DBAttachment2.

- CAbstract Diese Komponente erstellt lediglich eine cconfig.json Datei an dem unter option → confPath=Datei angegebenen Ort, mit den zugehörigen Verbindungen.

- DBAttachment2 Hierbei erhält die neue Komponente eigene Tabellen, entsprechend dem Muster attachment_Komponentenname.

2.4 Beispiel

```
{
  "name":"SampleA",
  "version":"1.0",
  "type":"clone",
  "base":"SampleB",
  "baseURI":"/sampleA",
  "option":"confPath=Assistants/sampleA_cconfig.json",
  "links":[
    {"name":"request","target":"","links":[]}
  ]
}
```

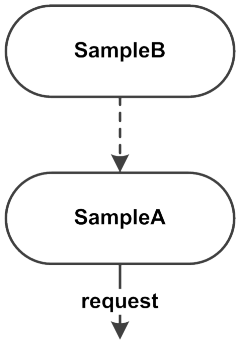

Abbildung 2.2: SampleA ist eine Kopie von SampleB, wobei Sample A einen zusätzlichen Ausgang request definiert

3 Aufrufdefinitionen

Sie können anderen Komponenten und Entwicklern Aufrufe anbieten, welche in der Commands.json der entsprechenden Komponente definiert werden.

3.1 Aufbau

- name Der Name des aufrufbaren Befehls (sollte ausreichend beschreibend sein)
- callback Die aufzurufende Funktion der Komponentenklasse (eventuell kümmert sich die Komponente auch selbst um den Ablauf, jedoch könnte eine entsprechende Hilfsklasse diese Information nutzen)
- outputType Das Ausgabeformat des Aufrufs (eine leere Zeichenkette bestimmt eine leere Ausgabe der ein unbekanntes Ausgabeformat). Sie können hier eine der im Assistants/Structures/ Ordner definierten Strukturen namentlich angeben.
- inputType Das Eingabeformat des Aufrufs (eine leere Zeichenkette bestimmt eine leere Ausgabe der ein unbekanntes Ausgabeformat)
- method Die HTTP Aufrufmethode. Diese sollten im wesentlichen GET, POST, PUT und DELETE sein, aber auch andere sind denkbar.
- path Der Befehl zum Aufruf dieser Funktion (URI).

3.2 Beispiel

```
[
    {
        "name": "getUserCourses",
        "callback": "getMatch",
        "outputType": "Course",
        "inputType": "",
        "method": "GET",
        "path": "/course/user/:userid"
    },
    {
        "name": "addPlatform",
        "outputType": "Platform",
        "inputType": "Platform",
        "method": "POST",
        "path": "/platform"
    },
    {
        "name": "deletePlatform",
        "outputType": "Platform",
        "inputType": "",
        "method": "DELETE",
        "path": "/platform"
    },
    {
        "name": "getExistsPlatform",
        "callback": "getMatchSingle",
        "outputType": "Platform",
        "inputType": "",
        "method": "GET",
        "path": "/link/exists/platform"
    }
]
```