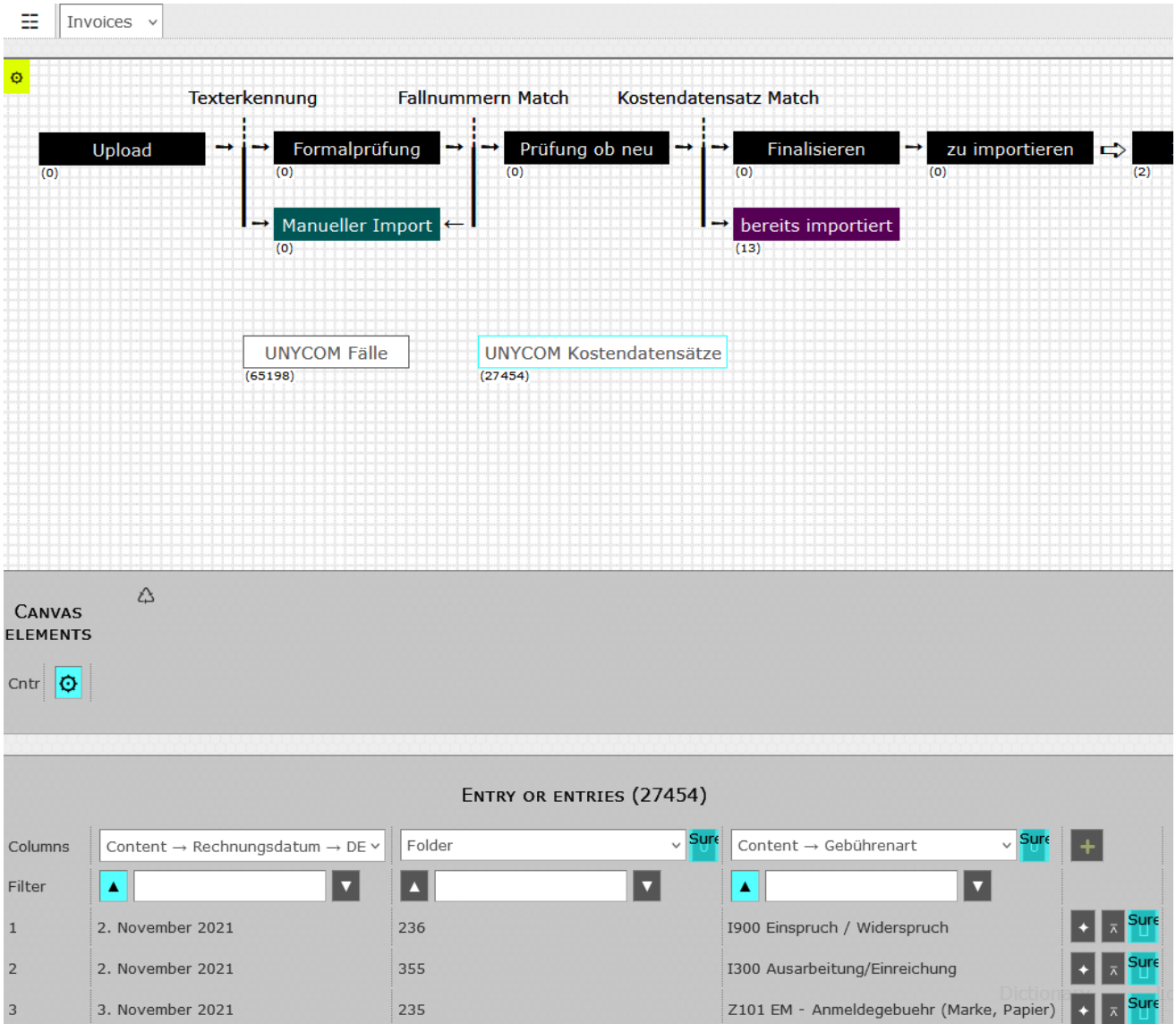
# I Datapool

Open Source Data Processing Framework




Autor: Carsten Wallenhauer
Stand: 2. April 2023

## I.1 Motivation

2019 hat meine Arbeitgeberin mit den Vorbereitungen zur SAP-Einführung begonnen hat. Bis dahin hatten wir für Jahrzehnte ein Eigengewächs, eine Software namens SIGMA, verwendet. Mit unserer Softwareausstattung hatten wir eine hohe Produktivität und Qualität erreicht.

In der Patentabteilung nutzten wir viele Jahre die Spezialsoftware PatOrg, sind dann aber vor fast 10 Jahren aber auf UNYCOM umgestiegen. Mit SAP und UNYCOM sind wir nun bei einer Softwareausstattung angelangt, die viele größere Unternehmen und Organisationen im deutschsprachigen Raum aufweisen.

Für Insider vielleicht nicht überraschend, haben diese Umwälzungen bisher kaum eine Verbesserung der Produktivität zur Folge gehabt. Im Gegenteil, in meinem Verantwortungsbereich sehe ich massive Produktivitätseinbußen. Insbesondere SAP verleitet mich oft zu einem Vergleich mit dem in Europa immer offensichtlicheren Problem der Überregulierung und komplexer Formalismen. Ursächlich scheint ein Streben nach absoluter Sicherheit und Dokumentation. Die Kosten dafür sind niedrige Produktivität und in Folge Inflation, Mangel an Fachkräften, mangelnde Konkurrenzfähigkeit und auf der menschlichen Seite, überarbeitete verzweifelte Kolleginnen und Kollegen.

Mit SAP hatte ich persönlich den Eindruck die Kontrolle zu verlieren. Die Meetings mit den SAP-Spezialisten waren geprägt von einer Art „SAP-Insidersprache“ bei der man sich stetig fragte, ob das wesentliche Ziel das Vermeiden einer Verhandlung auf Augenhöhe ist. Beeindruckend war der wiederholte Hinweis, dass wir in unserer erfolgreichen Vergangenheit alles falsch gemacht haben müssen. Meines Erachtens sind das Ergebnis praxisferne Prozesse, mit langen seriellen Genehmigungsketten und mangelndem Datenaustausch. Für das Erreichen der Produktivität aus der Vergangenheit werden wir in die Zukunft vertröstet.

### Die Kontrolle zurückerlangen, wieder effizient arbeiten...

Keines der großen Softwarepakete deckt alle Anforderungen einer Organisation ab. Entweder sind organisationsspezifische Anpassungen erforderlich oder Daten müssen außerhalb der großen Softwarepakete bearbeitet werden.

Organisationsspezifische Anpassungen haben den wesentlichen Nachteil nicht versionssicher zu sein. So kann ein Update zum Verlust oder zur Funktionsunfähigkeit der Anpassung führen. Das ist verständlich, letztendlich werden die großen Softwarepakete für die durchschnittliche Marktanforderung programmiert. Etwas Abhilfe schaffen vorgesehene Freiheiten für den Nutzer, die Software zu konfigurieren. An Grenzen stößt man, wenn ein Datenaustausch zwischen den großen Softwarepaketen erforderlich ist. Die Kosten sind typisch sehr hoch und der langfristige Mehrwert unsicher, da sich ggf. Anforderungen über die Zeit ändern oder neue Versionen der Basissoftware inkompatibel sind.

Die höchste Flexibilität ist dann erreichbar, wenn sich erforderliche Daten exportieren, außerhalb des Softwarepaketes verarbeiten und dann wieder importieren lassen. In der Praxis verbreitet ist die Bearbeitung der Daten mit Software zur Tabellenkalkulation (Excel und Co.). Im Einsatz sind aber auch Bots. D.h. Software, die sich wie ein Nutzer verhält, beispielsweise sich als Nutzer in zwei

Softwarepaketen anmeldet, Daten auf der einen Seiten entnimmt, prozessiert und dann in das andere Softwarepaket eingibt. Eine weitere Alternative sind sogenannte Low-Code-Plattformen die ich zwischen der Tabellenkalkulation und der aufgabenspezifischen Programmierung sehe. Der Mehrwert sind auslöserbasierte (durch Event oder sonstige Trigger) automatische Datenverarbeitung.

Für größere Organisationen lohnt es sich vermutlich eigene Softwareentwickler einzusetzen, die aufgabenspezifische Lösungen programmieren und pflegen.

Datapool ist für alle gedacht, die organisationsspezifische Datenverarbeitungslösungen brauchen. Die Basis bilden weit verbreitete Datenformate (Listen, pdf, JSON, XML) und Kommunikation via E-Mail und FTP. Die Basis ist Open Source, sodass jeder Ideen, Lösungen im Ganzen oder in Teilen nutzen und weiterentwickeln können.

## I.2 Basis

Das Framework ist in PHP geschrieben und kann lokal, in einem Intranet oder im Internet gehostet werden. Das Framework benötigt eine Datenbank, im Standard MySQL, Servertyp MariaDB, Zeichensatz UTF-8 mit einem zugehörigen Datenbanknutzer.

**Festlegungen:**

- ➔ Soweit möglich kommt durchgängig UTF-8 für den Zeichensatz zur Anwendung.
- ➔ Alles ist ein Datenpobjekt, eine E-Mail, ein Bild, ein Viedeo, ein Forumeintrag, eine csv-Datensatz oder auch nur eine Zeile aus einem csv-Datensatz.
- ➔ Einzelne Dateien eines zip-Archives sind jede für sich ein Datenobjekt.
- ➔ Ein Datenobjekt ist ein “Entry” in einer Tabelle der Datenbank.
- ➔ Einem “Entry” kann genau eine Datei mit der “EntryId” als Dateinamen zugeordnet sein.
- ➔ Daten eines “Entry” werden entweder in der Spalte “Content” als JSON kodiertes Array oder/und in der zugeordneten Datei gespeichert.
- ➔ Metadaten werden in der Spalte “Params” als JSON kodiertes Array gespeichert.
- ➔ Jeder “Entry” hat eine maximale Lebenserwartung, die in der Spalte “Expires” festgelegt ist.
- ➔ Jeder Datenbanktabelle (z.B. user) ist ein PHP-Skript (z.B. User.php) und eine Klasse (z.B. User) mit gleichem Namen zugeordnet, wobei der Name der Datenbanktabelle aus Kleinbuchstaben besteht.

### I.2.1 Installation

Datapool ist unter https://github.com/SourcePot/datapool verfügbar und kann mit Packagist installiert werden. So lässt sich lokal unter MS Windows das Framework durch folgendes Kommando erstellen:

```
composer create-project sourcepot/datapool <TARGET DIRECTORY>
```

Das <TARGET DIRECTORY> ist so zu wählen bzw. zu erstellen, dass es über Lokalhost oder den Server aus dem Netzwerk zugänglich ist. Bei Nutzung von XAMPP kann das Framework in einem neuen Unterverzeichnis „datapool“ installiert werden.

```
..\xampp\htdocs\datapool\
```

Durch die Installation wird die Verzeichnisstruktur erzeugt, einschließlich des Einstiegspunktes für den Web-Browser. Durch den Browser ist das Framework unter folgender Adresse zugänglich:

```
http://localhost/www-workspace/datapool/src/www/
```

Wobei das Unterverzeichnis „src\www\“ das Verzeichnis ist, auf dass beispielsweise im Intra- oder Internet die Domain zeigt und die Startseite „index.php“ über den Web-Browser aufruft.

Figur 1 zeigt ein Beispiel für Hosting-Einstellungen unter Verwendung eines virtuellen Servers, der mit Plesk administriert wird. Die Zugriffsrechte für alle Dateien im Verzeichnis „www\“ einschließlich der Dateien in den Unterverzeichnissen müssen über das Netzwerk lesbar für den Nutzer lesbar sein. Die Unterverzeichnisse „www\media\“ und „www\tmp\“ müssen lesbar und ausführbar sein.


*Figur 1: Dokumentenstamm für dem Zugang aus dem Internet*

## I.2.2 Konfiguration

**Wichtig:** Nach der Installation müssen einige grundsätzliche Einstellungen vorgenommen werden. Dazu wird das Framework das ersten Mal per Web-Browser aufgerufen. Der erste Aufruf endet zwar in einer Fehlermeldung, aber es werden noch erforderliche Verzeichnisse und Konfigurationsdateien erzeugt. Die Fehlerursache kann den Fehler-Protokolldateien im Debugging-Verzeichnis siehe Tabelle 1 entnommen werden.

Es muss nun eine Datenbank mit zugehörigem Datenbanknutzer erstellt werden. Als Datenbankkollation (collation) sollte „utf8_unicode_ci“ gewählt werden. Das Framework wird sich unter Nutzung des Datenbankbenutzers mit der Datenbank verbinden. Diesem Nutzer müssen genügend Rechte gegeben werden, um Datenbanktabellen zu erstellen, zu ändern und zu löschen.

Datenbankname, Nutzername und -passwort (Zugangsdaten) müssen zwischen Datenbank und Framework abgestimmt sein. Solche Zugangsdaten und andere Konfigurationen sind im Verzeichnis „src/setup/“ zu finden, siehe Tabelle 1. Die Zugangsdaten für die Datenbank sind in der Datei „src/setup/Database/connect.json“ gespeichert. Es handelt sich um ein im JSON-Format gespeichertes Array.

```
{"Read":32768,"Content":
{"dbServer":"localhost","dbName":"***","dbUser":"***","dbUserPsw":"***"},"Type":"array","
```

```
Date":"{{NOW}}","Write":32768,"Owner":"SYSTEM","Privileges":1}
```

Die mit „*******“ gekennzeichneten Felder müssen mit den korrekten Zugangsdaten gefüllt werden.

Das Framework benötigt nun einen Admin-Nutzer. Dieser Admin-Nutzer wird als Datensatz in der Datenbanktabelle „user“ eingefügt. Sofern die Datenbankverbindung vorliegt und das Framework durch Aufruf der Webpage aufgerufen wird und noch kein Admin-User gefunden wird, erzeugt das Framework selbst ein erstes Admin-Nutzerkonto. Die Anmeldedaten dieses initialen Admin-Nutzerkontos werden in der Datei „src\setup\User\initAdminAccount.json“ gespeichert:

```
{"Content":{"Admin email":"admin@datapool.info","Admin
password":"******"},"Type":"array","Date":"2023-03-31
20:24:19","Read":32768,"Write":32768,"Owner":"SYSTEM","Privileges":1}
```

Mit diesen Daten ist Daten ist die Anmeldung als Administrator auf der Webpage möglich. Als E-Mail-Adresse wird die für den Web-.Admin in der Konfiguration gespeicherte Adresse verwendet. Die Konfiguration ist in der Datei „src\setup\Backbone\init.json“ zu finden, z.B.:

```
{"Content":{"pageTitle":"Datapool","pageTimeZone":"Europe\/Berlin","mainBackgroundImageFi
le":false,"loginBackgroundImageFile":"main-
login.jpg","iconFile":"LaIsla2.ico","charset":"utf-8","cssFiles":["jquery-ui\/jquery-
ui.min.css","jquery-ui\/jquery-ui.structure.min.css","jquery-ui\/jquery-
ui.theme.min.css","dynamic.css"],"jsFiles":["jquery\/jquery-3.6.1.min.js","jquery-ui\/
jquery-
ui.min.js","dark.js"],"emailWebmaster":"admin@datapool.info"},"Type":"array","Date":"2023
-03-31 20:21:38","Read":32768,"Write":32768,"Owner":"SYSTEM","Privileges":1}
```

### I.2.3 Struktur

Der Sourcecode des Frameworks befindet sich im Unterverzeichnis „src\“. Daneben gibt es die Verzeichnisse „docs\“ und „vendor\“. Das Verzeichnis „vendor\“ wird dynamisch von Composer bei der Installation erzeugt und enthält externe Komponenten wie den pdf-Parser, die vom Framework eingebunden werden. Tabelle 1 zeigt die Struktur.

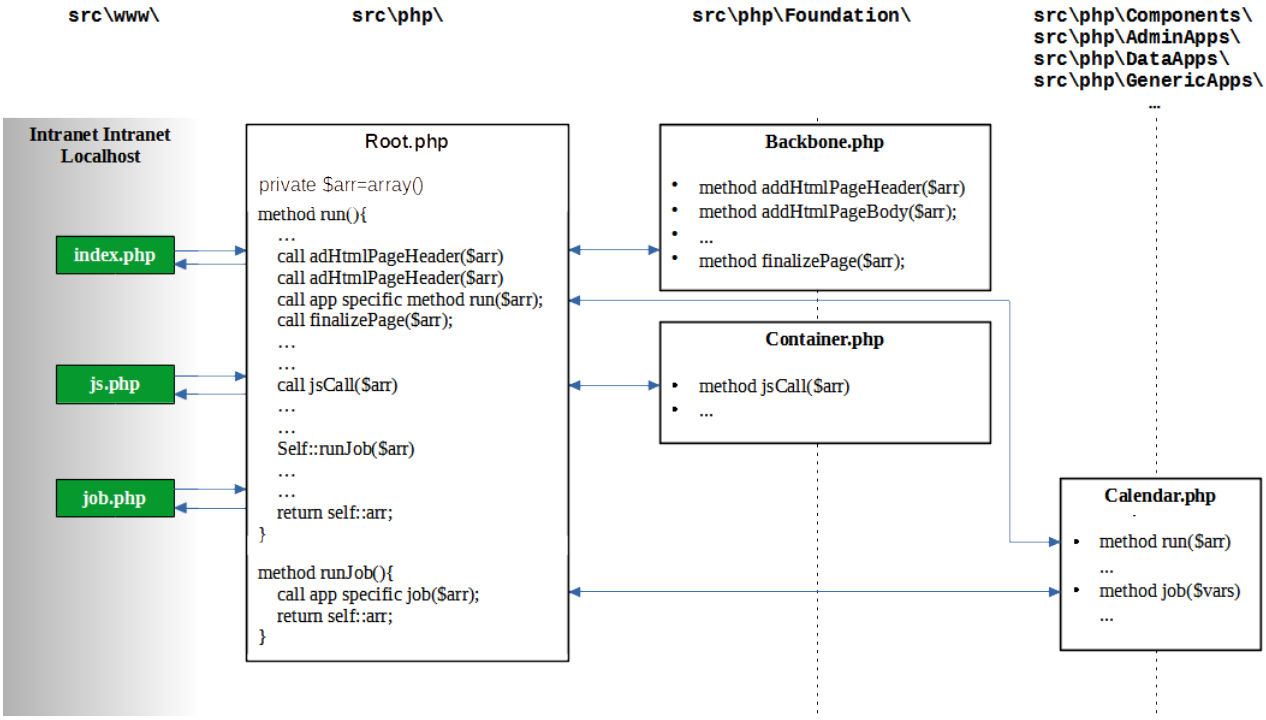
| | | |
|---|---|---|
| **docs** | | Hier findet sich Dateien zur Dokumentation |
| **src** | | Hier ist das Framework gespeichert |
| | **debugging** | Hier finden sich Fehlerprotokolle in Form von *.json Dateien |
| | **filespace** | Jedem Eintrag in der Datenbank kann eine Datei zugeordnet werden. Hier finden sich diese Dateien in Unterverzeichnissen, die den Namen der zugehörigen Datenbanktabelle tragen. Der Dateiname setzt sich zusammen aus der EntryId des Datenbankeintrags und dem Suffix „.file“. |
| | **fonts** | Hier finden sich Schriftfonts |
| | **fpdf** | Hier finden sich externe PHP Bibliotheken FPDF zur Erzeugung von pdf-Dateien. |
| | **ftp** | Verzeichnis zum Datenaustausch |

| | |
|---|---|
| **php** | Kern des Frameworks, die wesentlichen PHP Skripte |
| **setup** | Konfigurationsdateien des Frameworks im JSON Format |
| **www** | Dokumentenstamm für den Zugriff aus dem Netzwerk |
| **vendor** | Hier finden sich alle durch Composer eingebundenen externe Komponenten |

*Tabelle 1: Verzeichnisstruktur*

Neben den genannten Verzeichnissen finden sich die Composer-Konfigurationsdatei „composer.json“, die Lizenz un die Basisdokumentation in der Datei „README.md“.

Aus dem Netzwerk ist das Framework über drei PHP-Skripte möglich: index.php, job.php und js.php. Die dem Nutzer zugängliche Webpage wird beim Aufruf von „index.php“ erzeugt. Der Server sollte eingerichtet sein, das PHP-Skript „job.php“ periodisch aufzurufen (beispielsweise 1-Mal pro Minute). Dieses Skript verwaltet periodische jobs im Framework, z.B. das Löschen abgelaufener Datenbankeinträge. Das PHP-Skript „js.php“ dient der auf dem Clientcomputer laufenden JS-Skript „dark.js“ als Ziel für AJAX-Anfragen. Die Skripte „index.php“ und „job.php“ beantworten Anfragen mit HTML und das Skript „js.php“ mit JSON. Figur 2 zeigt die Kommunikation beim Aufruf der genannten Skripte.




*Figur 2: Struktur der wesentlichen PHP-Skripte des Frameworks*

Das Objekt der Klasse Root wird durch eines der drei PHP-Skripte “index.php”, “job.php” bzw. “js.php” erzeugt. Durch den Konstruktor dieses Objekts wird die private Eigenschaft $arr (vom Typ Array) initialisiert, die im Weiteren an alle anderen Klassen durch den jeweiligen Konstruktor und jeweilige init()-Methoden weitergegeben und ergänzt wird. Final enthält $arr['page html'] den

Rückgabe-String, der von dem PHP-Skript “index.php”, “job.php” bzw. “js.php” an den Web-Browser übermittelt wird.

Nach der Initialisierung enthält $arr die in folgeder Tabelle dargestellten Werte:

| **Schlüssel** | **Sub-Schlüssel** | **Struktur** |
|---|---|---|
| 'registered methods' | 'init' | array( class=>array(‘class’=>,’method’=>’’), ... class=>array(‘class’=>,’method’=>’’), ) |
| | 'job' | array( class=>array(‘class’=>,’method’=>’’), ... class=>array(‘class’=>,’method’=>’’), ) |
| | 'run' | array( class=>array(‘class’=>,’method’=>’’), ... class=>array(‘class’=>,’method’=>’’), ) |
| | 'unifyEntry' | array( class=>array(‘class’=>,’method’=>’’), ... class=>array(‘class’=>,’method’=>’’), ) |
| | 'dataProcessor' | array( class=>array(‘class’=>,’method’=>’’), ... class=>array(‘class’=>,’method’=>’’), ) |
| 'source2class' | Tabellenname | Zugeordnete Klasse |
| Klasse mit Namespace | Alle Objekte des Frameworks sind unter ihrer vollständigen Kalssenbezeichnung als Schlüssel in $arr gespeichert. | |

*Tabelle 2: Verzeichnisstruktur*

Nachdem die Initialisierungen abgeschlossen sind, ruft das jeweilige Skript “index.php”, “job.php” bzw. “js.php” die Methode run() des Root-Objektes auf. In Abhängigkeit vom aufrufenden Skript wird mit “Backbone.php” eine Webpage erzeugt oder mit “Container.php” eine JSON-Antwort auf eine JS-Anfrage oder mit der Methode runJob() eine Methode job() einer Zielklasse aufgerufen, die eine HTML-Antwort erzeugt.

# I.3 Jobs

## I.3.1 Jobs Grundlagen

Wenn das PHP-Skript „job.php“ durch beispielsweise einen CRON-Job aufgerufen wird, ermittelt das Skript welche job-Methode einer App am längsten überfällig ist. Diese job-Methode wird aufgerufen und das Ergebnis an „job.php“ übergeben. Mit welcher Periodizität die methode job($vars) einer Klasse aufgerufen werden soll, wird in Sekunden im Datenbankeintrag siehe Tabelle 4, Column „Content“ und Schlüssel „Min time in sec between each run“ vorgegeben. Die job-Methode wird mit dem Argument $vars aufgerufen und dieses Argument wird zum Abschluss auch zurück übergeben. Die in $vars gespeicherten Daten werden als Setting in der Datenbanktabelle „settings“ gespeichert, siehe Tabelle 3.

| **Column** | **Wert** |
|---|---|
| Group | Job processing |
| Folder | Var space |
| Name | Class, e.g. „SourcePot\Datapool\Processing\ CanvasProcessing“ |
| Type | array vars |
| Content | $vars = array(,statistis‘=>array(),…,) |

*Tabelle 3: Klassenspezifischer Datenbankeintrag zum ausgeführten Job*

Das Timing aller Jobs wird im Datenbankeintrag der Tabelle „setting“ mit dem Namen „Timing“ gespeichert.

| **Column** | **Wert** | | |
|---|---|---|---|
| Group | Job processing | | |
| Folder | All jobs | | |
| Name | Timing | | |
| Type | array setting | | |
| Content | Class, e.g. „SourcePot\Datapool\ Processing\CanvasProcessing“ | Last run | … |
| | | **Min time in sec between each run** | … |
| | | Last run time consumption [ms] | … |
| | | class | … |
| | | method | … |

*Tabelle 4: Datenbankeintrag zur Sicherung der Einstellungen zu allen Jobs*

## I.4 Datenverarbeitung

### I.4.1 DataApp Jobs