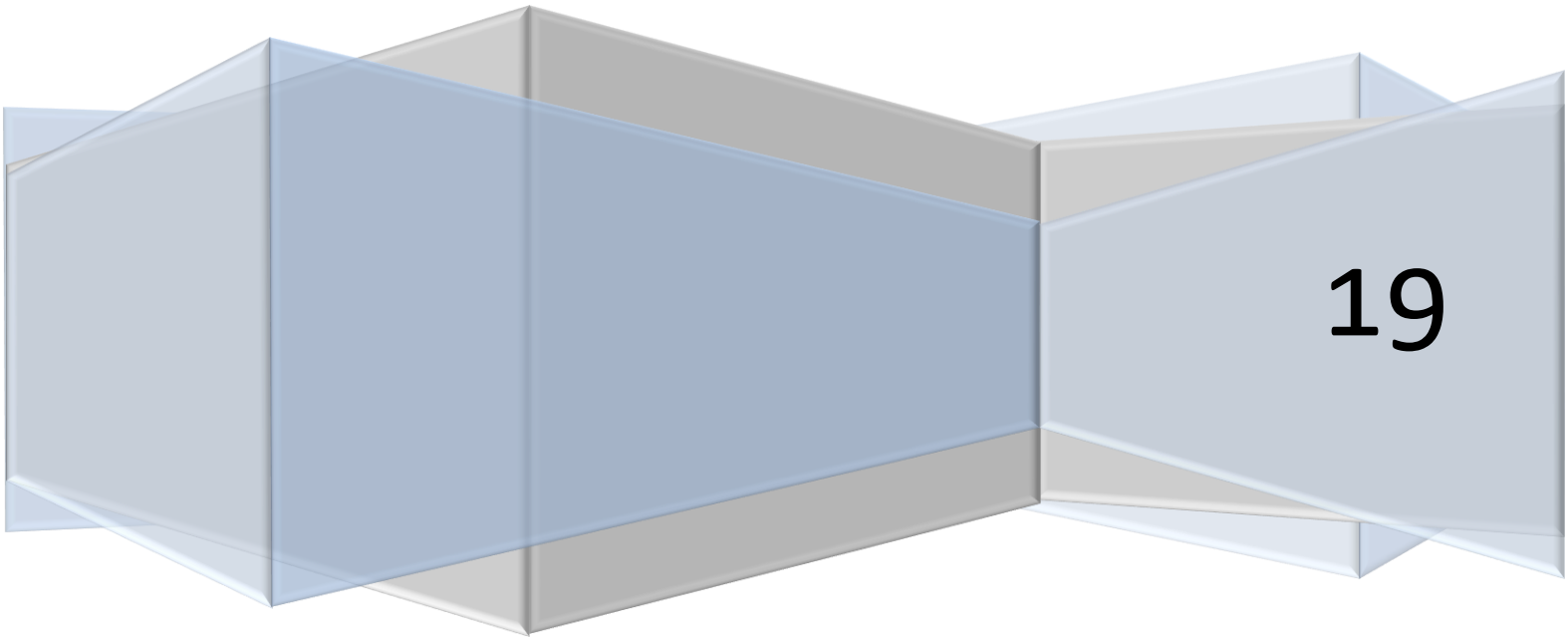
Konvertieren

Mit Sownloader Web

Marvin Klein



19

Inhalt

Vorwort

Ursprünglich war für dieses Projekt eine Zeiterfassung für Windows Programme geplant. Leider hat sich während der Bearbeitung des Projektes ergeben, dass die zuvor genannten Funktionalitäten nicht innerhalb des gegeben Zeitfensters erfüllt werden können, da dafür eine viel umfangreichere Einarbeitung in die Thematik notwendig gewesen wäre. Namentlich wäre hier z.B. die Windows Management Instrumentation (WMI)-Query zu nennen, über welche die Daten der Prozesse ausgewertet werden sollten. Darüber hinaus habe ich während des Projektes festgestellt, dass ich Projekte noch besser vorbereiten muss, da während der Programmierung immer wieder Fragen aufgetaucht sind, die im Voraus hätten klar sein müssen. Ein Beispiel wäre hier z.B. der Zusammenhang der Klasse Profile mit anderen Klassen.

Aufgrund der zuvor genannten Probleme, habe ich mich beschlossen ein anderes Projekt durchzuführen, welches auch noch innerhalb des Zeitraumes erledigt werden kann.

Einführung zum Mittelstufenprojekt

In diesem Projekt geht es um eine neue Funktion für den Webservice Sownloader. Diese Funktion erweitert Sownloader Web dahingehend, dass die heruntergeladenen Aufnahmen direkt Serverseitig in eine MP3 Datei umgewandelt werden können. Die Umwandlung findet mithilfe des Programms FFMPEG statt. Dieses muss mit den entsprechenden Parametern aufgerufen werden, damit der Input richtig umgewandelt werden kann. Da dieses Programm auf einem Webserver laufen soll, wird es in der Programmiersprache PHP in der Version 7.2 angefertigt. Das Front-End wird mit HTML und PHP zusammengestellt.

Hintergrund zum Mittelstufenprojekt

Sownloader ist 2014 aufgrund von Smule entstanden. Smule produziert bereits seit 2008 social Music Apps. Diese sollen es jeden Menschen ermöglichen Musik zu erschaffen und diese mit der Welt zu teilen. Die selbstgespielten/gesungenen Lieder können nun über die Apps auf den Servern von Smule hochgeladen werden, wo Sie dann die ganze Welt anhören kann. Die Problematik dabei ist, dass die Aufnahmen automatisch nach 365 Tagen gelöscht werden, sofern der Nutzer kein Premium-Abo gekauft hat. Sownloader bietet deshalb die Möglichkeit die Aufnahmen von den Servern herunterzuladen, sodass man diese auch nach 365 Tagen noch mit seinen Freunden, Famile und der Welt teilen kann.

Warum wird dieses Projekt durchgeführt?

Dieses Feature wird in Sownloader Web implementiert, da sich bereits mehr als 2000 User dieses Feature gewünscht haben. Eine Implementation des Features könnte dem Service dabei helfen, sich deutlich von der Konkurrenz hervorzuheben, da diese solch ein Feature nicht anbieten. Ferner bietet die Umwandlung in MP3 den User viele Vorteile, da z.B. auch Radiostationen Sownloader Web einsetzen und die zum Teil nur mit MP3 Dateien arbeiten können.

Ziele dieses Projekts

Die Ziele dieses Projektes sind die Einbindung des Umwandlungsfeatures (Primär). Als sekundäres Ziel sollen die heruntergeladenen MP3-Dateien statistisch erfasst werden.

Was muss beachtet werden?

1. Konvertierte Dateien müssen regelmäßig vom Server gelöscht werden, da der Speicherplatz des Servers auf 10GB beschränkt ist. Ein Livetest hat ergeben, dass dieser nach etwa 3 Stunden vollständig belegt ist. Daher wird noch zusätzlich ein Cronjob erstellt, welcher jede Stunde ausgeführt wird und alle Dateien, welche älter als fünf Minuten sind löscht. Eine Überprüfung auf fünf Minuten wird deshalb vorgenommen, weil sich ggf. noch Dateien im Ordner befinden, welche derzeit noch konvertiert werden. Würden diese nun einfach gelöscht werden, dann würde der Prozess beim User abbrechen.
2. Es muss sichergestellt werden, dass nur für die Dateiendung MP3 auch konvertiert wird und nicht für die anderen Download Optionen (M4A und MP4).
3. Es soll nachvollzogen werden können, wie viele und welche Lieder als MP3 heruntergeladen wurden. Dafür muss die bestehende Datenbankstruktur erweitert werden. Die Auswertung soll später in den Statistiken der Webseite abrufbar sein.
4. Es muss einen zusätzlichen Hinweis geben, dass das Feature sich noch In der BETA-Phase befindet und deswegen eventuell nicht alle Aufnahmen funktionieren.
5. Konvertieren als MP3 darf nicht für mobile Geräte zur Verfügung stehen, da die Dateien nicht heruntergeladen werden können.
6. FFMPEG muss am Server installiert sein

Durchführung des Projektes

Installation von FFMPEG

Im ersten Schritt muss FFMPEG am Server installiert werden. Dazu verbinde ich mich zunächst per SSH mit dem Server. Dazu werden folgende Befehle im Root-Verzeichnis ausgeführt.

```
WGET HTTPS://JOHNVANSICKLE.COM/FFMPEG/BUILDS/FFMPEG-GIT-AMD64-STATIC.TAR.XZ
TAR -XVF FFMPEG-GIT-AMD64-STATIC.TAR.XZ
MV ~/FFMPEG-GIT-AMD64-STATIC ~/FFMPEG
RM FFMPEG-GIT-AMD64-STATIC.TAR.XZ
```

Anpassung der Datenbank

Damit die MP3-Downloads in der Datenbank gezählt werden können, muss die bestehende Struktur um eine neue Spalte erweitert werden. Dazu wurde am Server folgender SQL-Befehl ausgeführt:

```
ALTER TABLE `sownloader_web_urls` ADD COLUMN `downloads_mp3` INT DEFAULT 0
```

Diese Spalte zählt alle MP3-Downloads hoch. Das Update geschieht später durch die Downloader Klasse. Um die Gesamtanzahl der MP3 Downloads zu erhalten kann demnach der SQL-Befehl

```
SELECT SUM(`downloads_mp3`) FROM `sownloader_web_urls`
```

Anpassung der Sownloader-Klasse

Im nächsten Schritt wurde die Klasse Sownloader um ein neues Attribut erweitert, welches für das Frontend den Downloadlink für die MP3 Datei beinhaltet. Darüber hinaus wurde der Zugriffsmodifier für das Attribut isMobile von privat auf public abgeändert, damit das Frontend weiß, welche

Downloadlinks mobil nicht angezeigt werden sollen. Zur Erkennung des Endgeräts wird die Library MobileDetect[1] eingesetzt.

Anpassung der Downloader-Klasse

Als nächstes wurde die Downloader-Klasse angepasst. Hier wurde eine neue Methode zum Herunterladen von MP3 Dateien hinzugefügt. Darüber hinaus wurde die Funktionalität zum Updaten der Downloads in eine extra Methode der Klasse ausgelagert und für MP3 Downloads erweitert.

```
public function downloadFileMp3($url, $name) {

        $ffmpeg = "/home/webpages/lima-city/marvinkleinmusic/ffmpeg/ffmpeg";
        $output = array();

        // Generate random filename
        $tmpFilename = $_SERVER["DOCUMENT_ROOT"] . "/tmp/" . $this-
>generateRandomString() . "mp3";

        exec($ffmpeg . " -i " . $url . " -f mp3 -b:a 320k -acodec libmp3lame "
. $tmpFilename . " 2>&1", $output, $return_code);

        if($return_code == 0)
        {
            header("Cache-Control: public");
            header("Content-Description: File Transfer");
            header('Content-Disposition: attachment; filename="' . $name .
'"');
            header("Content-Type: application/mp3");
            header("Content-Transfer-Encoding: binary");
            readfile($tmpFilename);
            unlink($tmpFilename);
            $this->databaseUpdate(true);
        }
        else
        {
            echo "Unknown error";
        }
    }
```

Über die Methode exec() wird die ffmpeg Datei auf dem Server aufgerufen. An FFMPEG werden nun folgende Parameter[2] übergeben:

-i [url]	Inputfile. Entweder eine Datei oder eine URL
-f mp3	Format des Ouputfiles
-b:a 320k	Audiobitrate 320 Kiloherz
-acodec libmp3lame	Zu benutzendes Codec zum Konvertieren
[OUTPUT]	Dateiname für den Output

[1] http://mobiledetect.net/
[2] https://www.ffmpeg.org/documentation.html

Als letztes wurde eine neue Parameterabfrage beim Aufruf der Datei hinzugefügt, welche den Downloadtyp erkennt und anhand der per GET übergebenen Daten den richtigen Downloadprozess beginnt.

```
if(isset($_GET['url']) && isset($_GET['name']) && $_GET['pkey'] &&
isset($_GET['ext']) && $_GET['ext'] != "mp3") {
    $downloader = new Downloader($_GET['pkey']);
    $downloader->downloadFile($_GET['url'], $_GET['name'], $_GET['ext']);
}
else if(isset($_GET['url']) && isset($_GET['name']) && $_GET['pkey'] &&
$_GET['ext'] == "mp3") {
    $downloader = new Downloader($_GET['pkey']);
    $downloader->downloadFileMp3($_GET['url'], $_GET['name']);
}
```

Mithilfe des übergebenen Werts des ext Parameters kann die passende Methode zugeordnet werden. Das Update der Datenbank geschieht bei einem erfolgreichen Download automatisch am Ende der jeweiligen Methoden. Dabei wird an die Methode ein bool übergeben, welcher angibt, ob es sich bei dem Update um einen MP3 Download handelt, oder nicht.

```
private function databaseUpdate($isMp3Download = false) {
        $sql = "UPDATE `sownloader_web_urls` SET `downloads` = `downloads` +
1, `lastDownload` = NOW() WHERE `performance_key` = '" . $this->connection-
>real_escape_string($this->performance_key) . "'";
        $this->connection->query($sql);
        $sql = "INSERT INTO `sownloader_web_downloads_today` VALUES('" .
$this->connection->real_escape_string($this->performance_key) . "', NOW(), '"
. md5($_SERVER['HTTP_X_FORWARDED_FOR']) . "') ON DUPLICATE KEY UPDATE
`download_date` = NOW()";
        $this->connection->query($sql);

        if($isMp3Download):
            $sql = "UPDATE `sownloader_web_urls` SET `downloads_mp3` =
`downloads_mp3` + 1 WHERE `performance_key` = '" . $this->connection-
>real_escape_string($this->performance_key) . "'";
            $this->connection->query($sql);
        endif;
    }
```

Anpassung des Front-Ends

Nachdem die Arbeiten im Back-End erledigt worden sind, können nun die Arbeiten am Front-End erledigt werden. Diese sind dank der Klassen sehr schnell erledigt. Mithilfe der MobileDetect Library kann nun mit den Methoden isAndroid() und isIos() abgerfagt werden, ob das derzeitige Gerät ein Mobilgerät ist.

Diese beiden Methoden wurden nun verknüpft, dass der Inhalt in dem Block nur angezeigt wird, wenn das Gerät nicht IOS und nicht Android ist. Das Ergebnis sieht nun wie folgt aus:

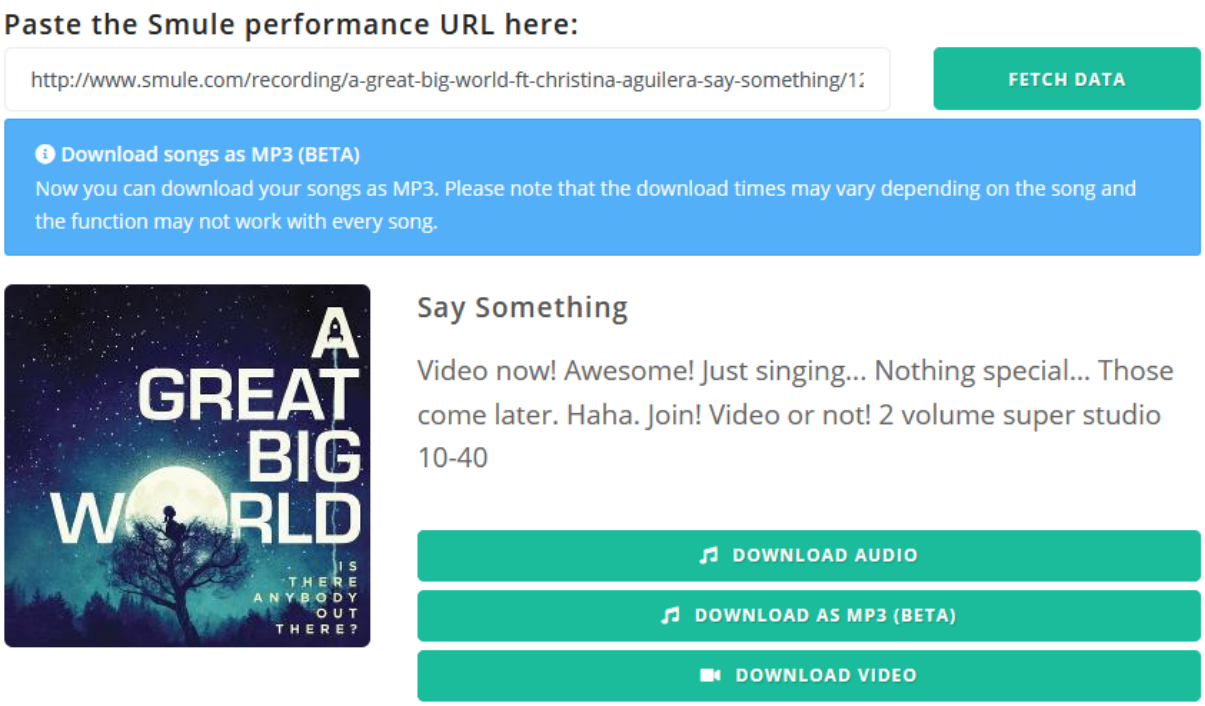


Abbildung 1 Sownloader Web Desktop

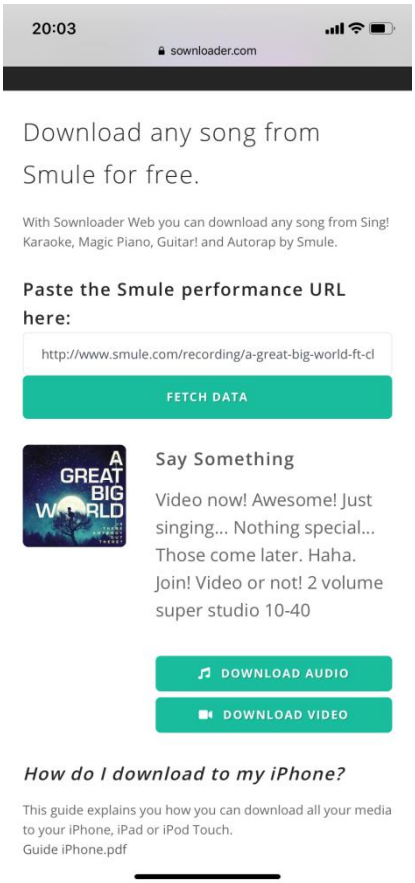


Abbildung 2 Sownloader Web Mobil

Erstellen des Cronjobs

Nun muss noch gewährleistet werden, dass die umgewandelten Dateien nicht auf dem Server liegen bleiben, da der Service nicht mehr funktionieren würde, wenn der gesamte Speicherplatz belegt ist.

Dazu wurde ein Cronjob (mp3_cleanup.php) im Verzechnis system/cron angelegt. Da die Dateien sich nur in einem Verzeichnis befinden können, muss das Verzeichnis nicht rekursiv durchgangen werden. Es reicht vollkommen aus, alle Dateien des angegeben Verzeichnis zu überprüfen.

Der Cronjob wurde nun in der Webverwaltung eingetragen, sodass er zu jeder Stunde zur dritten Minuten an jedem Tag in jedem Monat läuft.

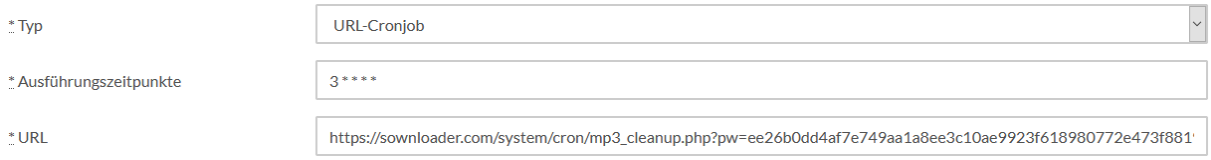


Abbildung 3 Cronjob in der Webverwaltung

Es wurde absichtlich 3 Minuten ausgewählt und nicht zur vollen Stunde, da zur vollen Stunde sehr viele Cronjobs auf dem Webserver ausgeführt werden. Dadurch, dass dieser ein wenig zeitversetzt ausgeführt wird, wird der Server nicht so stark belastet.

Statistiken

Zum Abschluss muss nur noch die Ausgabe der Downloadstatistik erledigt werden. Dazu wurde die Methode zum Abrufen der Downloadstatistiken mit der SQL Anfrage aus *Anpassung der Datenbank* angepasst.

Gibt es bereits Resultate des Projektes?

Ja, innerhalb der ersten zwei Tage wurden bereits über 250 Lieder erfolgreich konvertiert und heruntergeladen. Die Aktuelle Statistik lässt sich jederzeit über https://sownloader.com/stats abrufen.

Anlagen

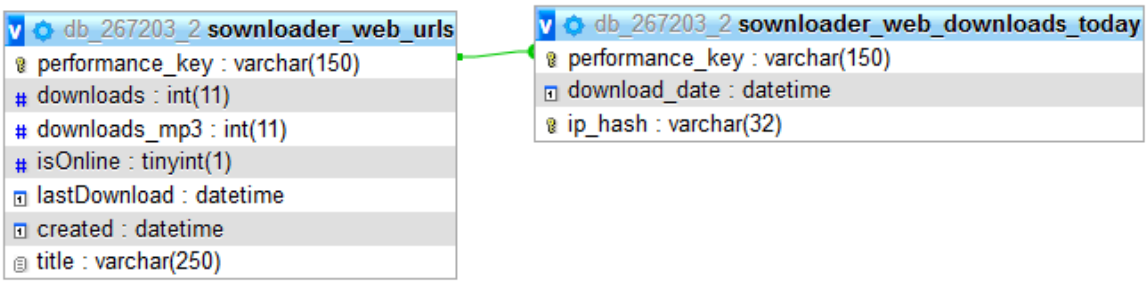


Abbildung 4 Datenbank Struktur

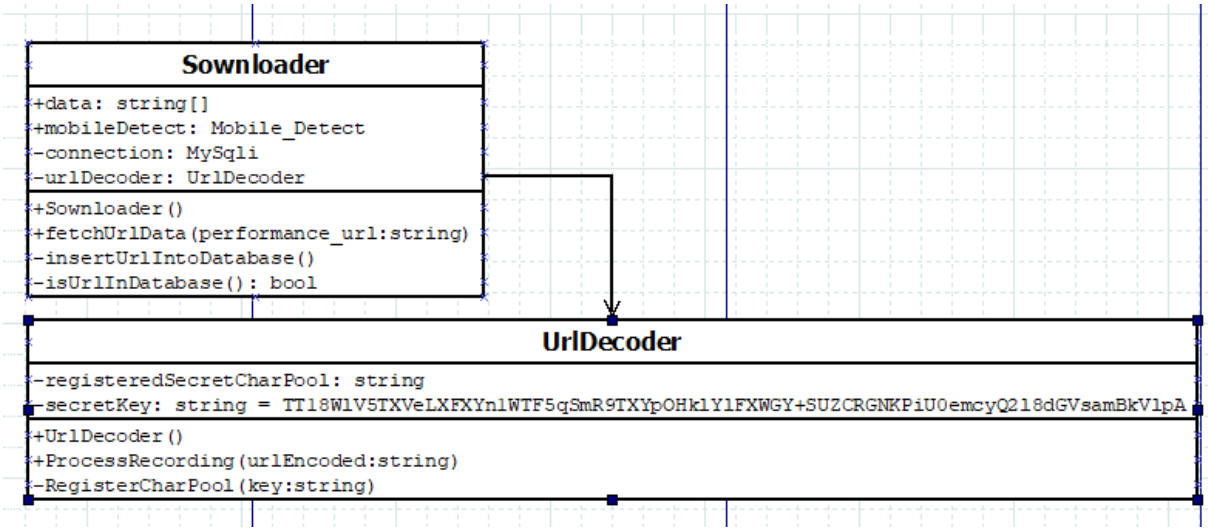


Abbildung 5 Klassendiagramm Sownloader & UrlDecoder

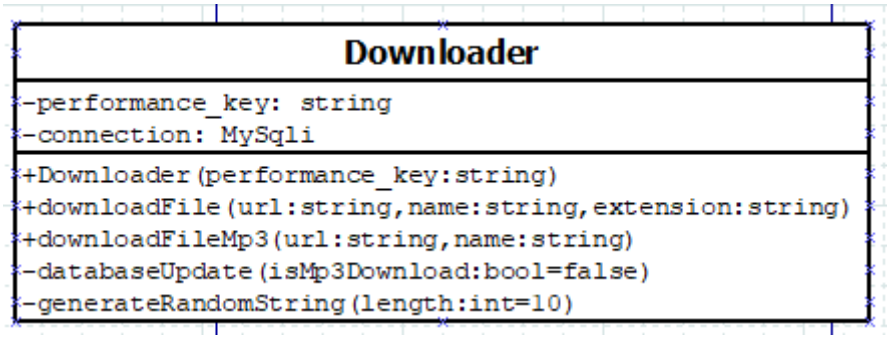


Abbildung 6 Klassendiagramm Downloader

Glossar

Cronjob	Eine Aufgabe die in einem angegeben Zeitraum von einem Server ausgeführt wird.
FFMPEG	Eine kostenfreie Library zum umwandeln von Mediadateien.
Zugriffsmodifier	Gibt an, welche Klassen auf ein Attribut zugreifen darf. Mögliche Werte sind hier z.B. private, public und protected.
Codec	Bezeichnet ein Algorythmenpaar, dass angibt, wie Signale digital kodiert und dekodiert werden.
SQL	Structured Query Language ist eine Sprache , die Befehle und Abfragen an Datenbanken ausführt.