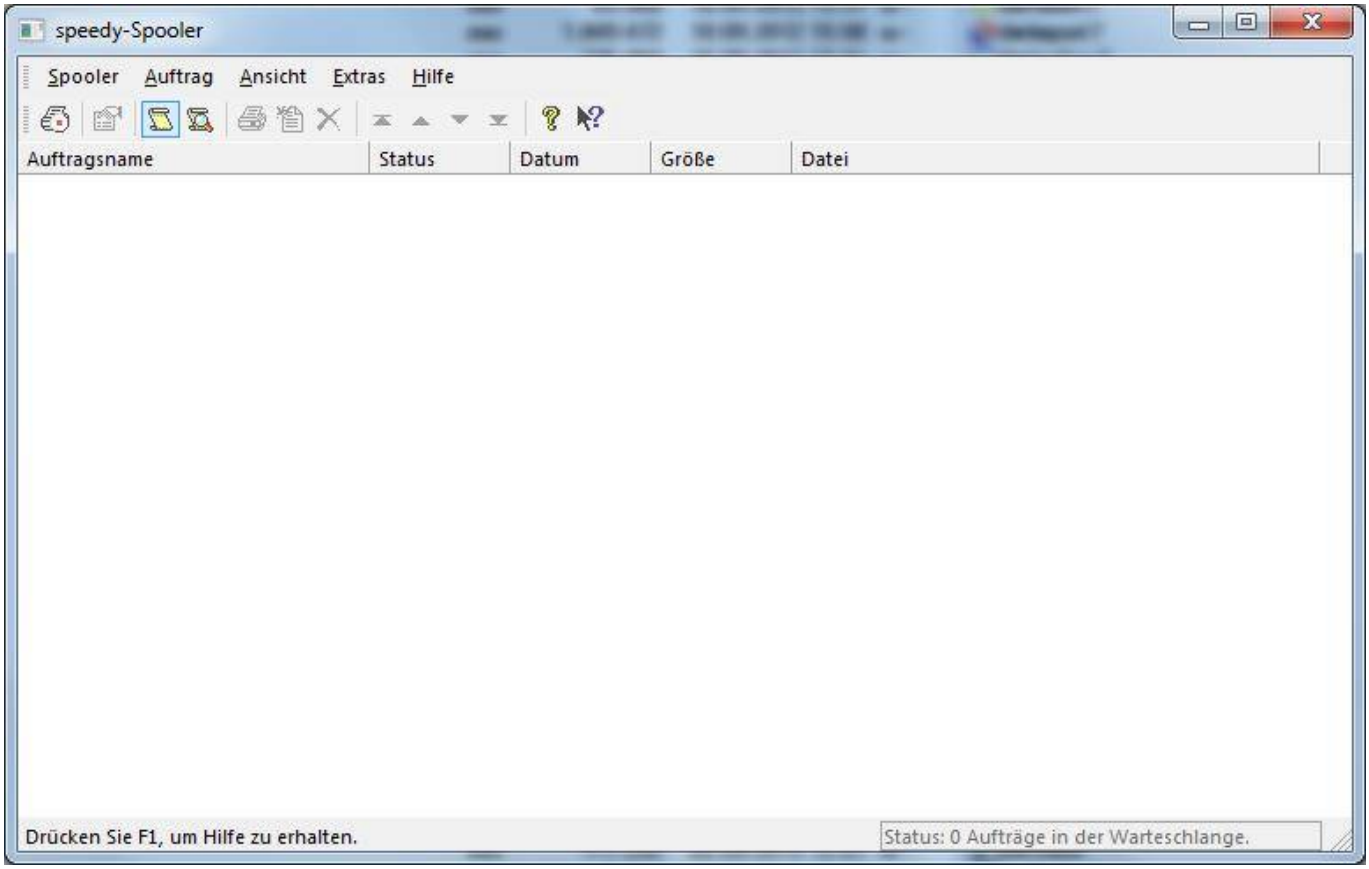
Spooler

Die speedy/PDM Spooler Anwendung überwacht ein oder mehrere Verzeichnisse und führt nach Eintreffen einer neuen Datei, in ein überwachtes Verzeichnis, einen definierten Befehl aus.
Es können beliebig viele Verzeichnisse überwacht werden. Die Überwachung kann verschiedene Dateiendungen oder Dateinamensmuster unterscheiden.



Lieferumfang

Datei	Beschreibung
bin32\dwSpool7.exe	Spooler Programm
bin64\dwSpool7.exe	
bin32\dwSpool.ini	Konfigurationsdatei
bin64\dwSpool.ini	

Konfiguration

Die Konfiguration des Spoolers erfolgt mit Hilfe einer Konfigurationsdatei im INI-Format. Die Konfigurationsdatei muss im gleichen Verzeichnis wie der das Spooler Programm liegen.

[SYSTEM]
Application = “dwSpool“
Version = 1

```
[SPOOLER]
TimeOut=500
WorkDir=.\spooler\
ErrorDir=.\error\
LogDir=.\log\

[WATCHDIR]
WatchDir1=.\import\

[WatchDir1.CSV]
Filter=art*.csv
Description=
Command=.\dwImport.exe -i “%1“ -t docpropimport -e “%2“ -l “%3“ -q
```

Schlüssel	**Beschreibung**
[SPOOLER]	
TimeOut	Wartezeit, in Millisekunden, zwischen erkennen einer neuen Datei und ausführen des entsprechenden Befehls.
ProcessPriority	Prozess Priorität. 0 := Idle 1 := Normal 2 := Hoch 3 := Echtzeit
WorkDir	Allgemeines Arbeitsverzeichnis. Nach dem Eintreffen einer neuen Datei, in ein überwachtes Verzeichnis, wird diese erstmals in das Arbeitsverzeichnis verschoben und anschließend von dort verarbeitet.
ErrorDir	Verzeichnis für fehlerhaft verarbeitete Dateien .
LogDir	Protokollverzeichnis.
ArchiveDir	Archivverzeichnis.
[WATCHDIR]	
WatchDir1	Zu überwachende Verzeichnisse
WatchDir2	
…	
[WatchDir1.EXT]	Für jedes zu überwachende Verzeichnis und die dazugehörige Dateierweiterung existiert ein eigener Konfigurationsabschnitt.
Description	Beschreibung des Filters.
Filter	Dateifilter der zu überwachenden Dateien. z.B.: Filter=“*.txt“ Oder Filter=“art*.csv“
WorkDir	Besonderes Arbeitsverzeichnis zur Abarbeitung der Dateien. [optional]

Schlüssel	Beschreibung
Command	Befehlszeile, die für jede Datei ausgeführt wird. Die Befehlszeile muss mind. einen Platzhalter zur Übergabe des Dateinamens beinhalten. Zusätzlichen stehen noch weitere Platzhalter zur Übergabe von Fehler- und Protokollverzeichnis zur Verfügung. z.B.: `Command=dwImport.exe "%1" "%2" "%3"` %1 := Arbeitsdatei %2 := Fehlerverzeichnis %3 := Protokollverzeichnis %4 := Archivverzeichnis
ShowWindow	Definiert die Anzeigeoption für das Befehlsfenster. 0:=SW_HIDE (standard) 1:=SW_SHOWNORMAL [optional]
UseAllEqual	Definiert, ob alle gleich benannten Dateien ebenfalls mit in das Arbeitsverzeichnis kopiert werden. [optional]
ErrorDir	Verzeichnis für fehlerhaft verarbeitete Dateien. [optional]
ArchiveDir	Archivverzeichnis. [optional]

Jobserver

Um einen Jobserver einzurichten wird der Spooler-Mechanismus genutzt. Jobs können nur abgearbeitet werden wenn die entsprechende Konfiguration erstellt wurde und die entsprechenden Module vorhanden sind. Es wird eine Transferdatei genutzt um die Jobs an den Jobserver zu übergeben. Diese Datei hat die Endung *.speedyjob. Wird ein Jobserver verwendet werden die Jobs auf dem Server ausgeführt.
Dies bedeutet:

- Bei Druckjobs muss das verwendete CAD System vorhanden sein
- Entsprechende Leistung (CPU, Arbeitsspeicher, Grafik, ...) muss auf dem Server vorhanden sein

Momentan verfügbare Jobs:

- jsPlot: Stapeldruck auf dem Jobserver
- jsRendition: Bildgenerierung/Vorschaugenerierung auf dem Jobserver

Ablauf des Jobserver

Der Jobserver überwacht ein Verzeichnis. Befindet sich in diesem Verzeichnis eine *.speedyjob-Datei wird diese eingelesen und die entsprechenden Einträge in der js_jobs Tabelle in der Datebank gemacht. Danach wird die speziell für diesen Jobtyp eingestellt .exe aufgerufen die dann den entsprechenden Druckjob ausführt.

Konfiguration

Um den Spooler als Jobserver zu Nutzen ist folgende Konfiguration in der dwSpool.ini zu machen:

[WATCHDIR]
WatchDirX = *„Pfad für die Ablage der Jobdateien“ (siehe auch [plot.jobserver.path])*

[WatchDirX.SPEEDYJOB]
Filter=*.speedyjob
Description=
Command=%1
ShowWindow = 1

Dies dient als Beispiel und kann ihrer Konfiguration angepasst werden.
Desweiteren werden 3 Tabellen in der Datenbank benötigt:

- js_jobs
- js_proc
- js_types

Diese 3 Tabellen nutzt der Jobserver um die Jobs abzuarbeiten.

In der **js_types** Tabelle werden die zu verfügbar stehenden Jobs definiert

Schlüssel	**Beschreibung**
types_id	ID des Typs
types_name	Name des Types. Beispiel: plot→Beschreibt den Druckjob
types_parallelcount	Beschreibt wieviele Prozesse von diesem Typ gleichzeitig gestartet werden können
types_maxtime	Maximale Zeit die ein Job brauchen darf(in Millisekunden). Nach dieser Zeit wird der Job beendet. Dies kann genutzt werden um aufgehängte Jobs neu starten zu können
types_killable	Gibt an ob nach abgelaufener maxtime der Prozess beendet werden kann oder nicht
types_maxrestart	Max Wert wie oft ein Prozess erneut gestartet werden kann
types_settinghead	gibt den Knoten der Eigenschaften an die zu diesem Job gehören. Beispiel: plot→ dadurch werden die Einstellungen plot.jobserver eingelesen und verwendet. *.jobserver.available gibt an ob der Jobserver aktiv ist. *.jobserver.path gibt an welcher Ordner überwacht werden soll.
types_exepath	Gibt an wo die EXE liegt die gestartet werden soll.
types_flag	Allgemeines Flag

Beispielkonfiguration eines Drucktyps:

types_id	types_name	types_parallelcount	types_maxtime	types_killable	types_maxrestart	types_settinghead	types_exepath	types_flag
2	plot	1	50.000	1	1	plot	[redacted]jsPlot.exe	0

Tabellen

Tabellendefinitionen der Jobserver Tabellen.

js_jobs

```
CREATE TABLE `js_jobs` (
    `jobs_id` INT(11) NOT NULL AUTO_INCREMENT,
    `jobs_type` VARCHAR(50) NULL DEFAULT NULL,
    `jobs_schedule` DATETIME NULL DEFAULT NULL,
    `jobs_prior` INT(11) NULL DEFAULT '0',
    `jobs_createdby` VARCHAR(50) NULL DEFAULT NULL,
    `jobs_createdat` DATETIME NULL DEFAULT NULL,
    `jobs_response` INT(11) NULL DEFAULT '0',
    `jobs_responseusr` VARCHAR(50) NULL DEFAULT NULL,
    `jobs_host` VARCHAR(50) NULL DEFAULT NULL,
    `jobs_time` DATETIME NULL DEFAULT NULL,
    `jobs_flag` INT(11) NULL DEFAULT '0',
    `jobs_path` VARCHAR(255) NULL DEFAULT NULL,
    `jobs_restarted` INT(11) NULL DEFAULT '0',
    PRIMARY KEY (`jobs_id`)
)
COLLATE='latin1_swedish_ci'
ENGINE=InnoDB
AUTO_INCREMENT=85
;
```

js_proc

```
CREATE TABLE `js_proc` (
    `proc_id` INT(11) NOT NULL AUTO_INCREMENT,
    `proc_jid` INT(11) NOT NULL,
    `proc_jsdtype` VARCHAR(50) NULL DEFAULT NULL,
    `proc_startat` DATETIME NULL DEFAULT NULL,
    `proc_endat` DATETIME NULL DEFAULT NULL,
    `proc_result` VARCHAR(255) NULL DEFAULT NULL,
    `proc_finished` INT(11) NULL DEFAULT '0',
    `proc_flag` INT(11) NULL DEFAULT '0',
    PRIMARY KEY (`proc_id`),
    INDEX `proc_jid` (`proc_jid`)
)
COLLATE='latin1_swedish_ci'
ENGINE=InnoDB
AUTO_INCREMENT=27381
;
```

js_types

```
CREATE TABLE `js_types` (
    `types_id` INT(11) NOT NULL AUTO_INCREMENT,
    `types_name` VARCHAR(50) NULL DEFAULT NULL,
```

```
    `types_parallelcount` INT(11) NULL DEFAULT NULL,
    `types_maxtime` INT(11) NULL DEFAULT NULL,
    `types_killable` INT(11) NULL DEFAULT '0',
    `types_maxrestart` INT(11) NULL DEFAULT '0',
    `types_settinghead` VARCHAR(50) NULL DEFAULT NULL,
    `types_exepath` VARCHAR(255) NULL DEFAULT NULL,
    `types_flag` INT(11) NULL DEFAULT NULL,
    PRIMARY KEY (`types_id`)
)
COLLATE='latin1_swedish_ci'
ENGINE=InnoDB
AUTO_INCREMENT=3
;
```