



Quickstart

Einstieg in das Job Scheduling
Februar 2009

Impressum

Software- und Organisations-Service GmbH
Giesebrechtstr. 15

D-10629 Berlin

Telefon (030) 86 47 90-0

Telefax (030) 8 61 33 35

Mail info@sos-berlin.com

Web www.sos-berlin.com

Letzte Aktualisierung: Dezember 2008

Inhaltsverzeichnis

1 Einleitung	4
1.1 Überblick über die Dokumentationen des Job Schedulers	4
2 Wie können Jobs erstellt und konfiguriert werden?	5
3 Einen Job mit dem Job Editor im Hot Folder einrichten	6
3.1 Vorbereitung	6
3.2 Durchführung	6
4 Einen einfachen Job einrichten	11
4.1 Voraussetzungen	11
4.2 Vorgehensweise	11
5 Einen einfachen Managed Job einrichten	13
5.1 Voraussetzungen	13
5.2 Vorgehensweise	14
6 Anwendungsfälle für Job-Ketten	21
6.1 Jobs in Reihe starten mit automatischer Fehlerbehandlung	21
6.2 Jobs in Reihe starten mit manueller Fehlerbehandlung	22
6.3 Jobs in Reihe starten mit automatischer Wiederholung im Fehlerfall	23
6.4 Jobs in Reihe starten mit manueller Wiederholung im Fehlerfall	24
6.5 Manuelles Überspringen von Jobs in einer Job-Kette	25
6.6 Manuelles Anhalten von Jobs in einer Job-Kette	26
6.7 Auf Verzeichnisänderungen reagieren	26
Anhang A: Beispiel 1: Zeitgesteuertes Ausführen eines Shell-Skripts	28
Anhang B: Beispiel 2: Ausführen eines Shell-Skripts per Verzeichnis-Überwachung	29
Anhang C: Beispiel 3: Ausführen eines PHP-Skripts	30
Anhang D: Beispiel 4: Programme ausführen	31
Glossar	32

1 Einleitung

Der Job Scheduler ist installiert und konnte gestartet werden. Was nun?

Die folgenden Beispiele sollen einen schnellen Einstieg in die Verwendung des Job Scheduler liefern. In diesem Quickstart verwendete relative Pfade beginnen im Installationsverzeichnis des Job Schedulers, welches gleichzeitig sein Arbeitsverzeichnis ist.

(siehe Kapitel Job Requirements (Seite 11)). Bei URLs ist `[host]` der Hostname des Rechners, auf dem der Job Scheduler installiert wurde.

Um den Job Scheduler eigenen Web Server aufzurufen, geben sie im Browser nach dem Hostnamen die Nummer des TCP Ports an.

Z.B. `http://localhost:4444/`

1.1 Überblick über die Dokumentationen des Job Schedulers

Installation und Konfiguration

- **Installation und Konfiguration (scheduler_installation)**
wir empfehlen die Durchsicht bevor Sie installieren und konfigurieren.
- **Quickstart zur Einrichtung von Jobs (scheduler_quickstart)**
Dieses Dokument
- **Referenzdokumentation (scheduler)**
Das Handbuch beschreibt Job-Konfiguration und Schnittstellen

Implementierung von Jobs mit dem Job Scheduler API

Diese Dokumentationen richten sich an Entwickler von Jobs, sie sind nicht erforderlich zum Einrichten automatisierter Starts von Programmen und Skripten.

- **API Dokumentation (scheduler_api)**
Die Dokumentation der Programmschnittstelle des Job Schedulers
- **Tutorial Job-Implementierung (scheduler_tutorial)**
Einführung in die Verwendung der Programmschnittstelle des Job Schedulers

Fortgeschrittene Themen

Diese Dokumentationen beschreiben komplexere Szenarien des Job Scheduling.

- **Tutorial Web Service Implementierung (scheduler_webservices)**
Erläutert die Konfiguration und Implementierung von Web Services für Ihre Jobs
- **Managed Job Scheduling (scheduler_managed_jobs)**
Die Dokumentation um Jobs in einer Datenbank zu verwalten
- **MySQL Job Scheduling (scheduler_managed_user_jobs)**
Erklärt die SQL-Schnittstelle des Job Schedulers für Statements und Prozeduren, die in der Datenbank analog zum Oracle Job Scheduler verwendet werden.

2 Wie können Jobs erstellt und konfiguriert werden?

Es gibt mehrere Varianten, wie eine Job erstellt und konfiguriert werden kann:

- Ein Job kann mit dem Job Editor angelegt und bearbeitet werden.
Hierzu wird eine bestehende Konfigurationsdatei im Job Editor geöffnet und bearbeitet, oder es wird eine neue Konfigurationsdatei angelegt.
- Ein Job kann mit einem beliebigen Texteditor angelegt und bearbeitet werden.
Hierbei werden die XML-Konfigurationsdateien "per Hand" verändert.
- Ein Job kann mit der Managed Jobs Oberfläche angelegt und bearbeitet werden.

Darüberhinaus kann man zwei verschiedene Arten der Scheduler-Konfiguration bearbeiten:

- **Die statische Konfiguration:**
Hierbei wird die Konfigurationsdatei `scheduler.xml` im Verzeichnis `[scheduler install path]/config` und optional weitere Konfigurationsdateien (siehe Basiskonfiguration), die in `scheduler.xml` eingebunden sind angepasst.
Wenn hier Änderungen gemacht werden, ist ein Neustart des Job Schedulers erforderlich.
Beim Neustart wird die Konfigurationsdatei `scheduler.xml` und alle abhängigen Dateien neu eingelesen.
- **Die dynamische Konfiguration:**
Hierbei wird der Inhalt des "Hot Folders", per default das Verzeichnis `[scheduler install path]/config/live` angepasst.
In diesem Verzeichnis wird jedem Job, jeder Job Kette und anderen Elementen jeweils eine Datei zugeordnet, die die gesamte Konfiguration dieses Elements beinhaltet.

3 Einen Job mit dem Job Editor im Hot Folder einrichten

Diese Anleitung beschreibt das Vorgehen zum Einrichten eines Jobs im Job Scheduler mit dem Job Editor, sowohl unter Windows als auch unter Unix.

Da die Änderungen im "Hot Folder" stattfinden, werden die Änderungen wirksam ohne dass der Job Scheduler neu gestartet werden muss.

3.1 Vorbereitung

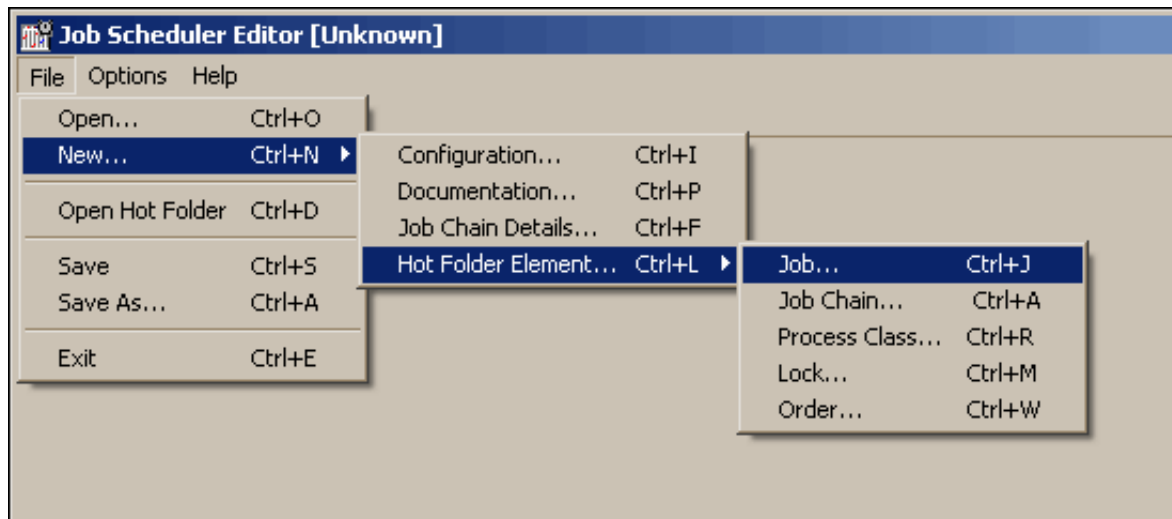
Der Job Scheduler soll gestartet sein. Die Änderungen werden also während des laufenden Betriebs gemacht.

3.2 Durchführung

Starten Sie den Job Editor mit dem Skript `jobeditor.cmd` (Windows) bzw. `jobeditor.sh` (Unix).

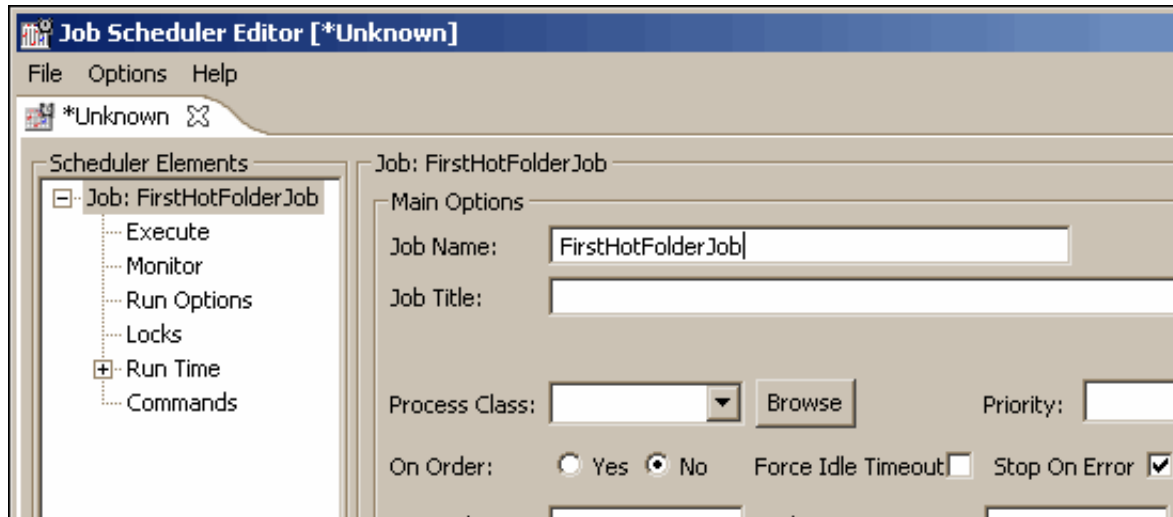
Dieses Skript befindet sich im Verzeichnis `[scheduler install path]/bin`.

Öffnen Sie im Editor `File->New...->Hot Folder Element...->Job...`



Es wird ein leeres Konfigurationselement für einen Job angelegt.

Tragen Sie ins Feld "Job Name" z.B. ein: `FirstHotFolderJob`



Wählen Sie danach den Unterabschnitt "Execute".

Es öffnet sich eine Maske, in der das Programm angegeben werden kann, das der Job ausführen soll.

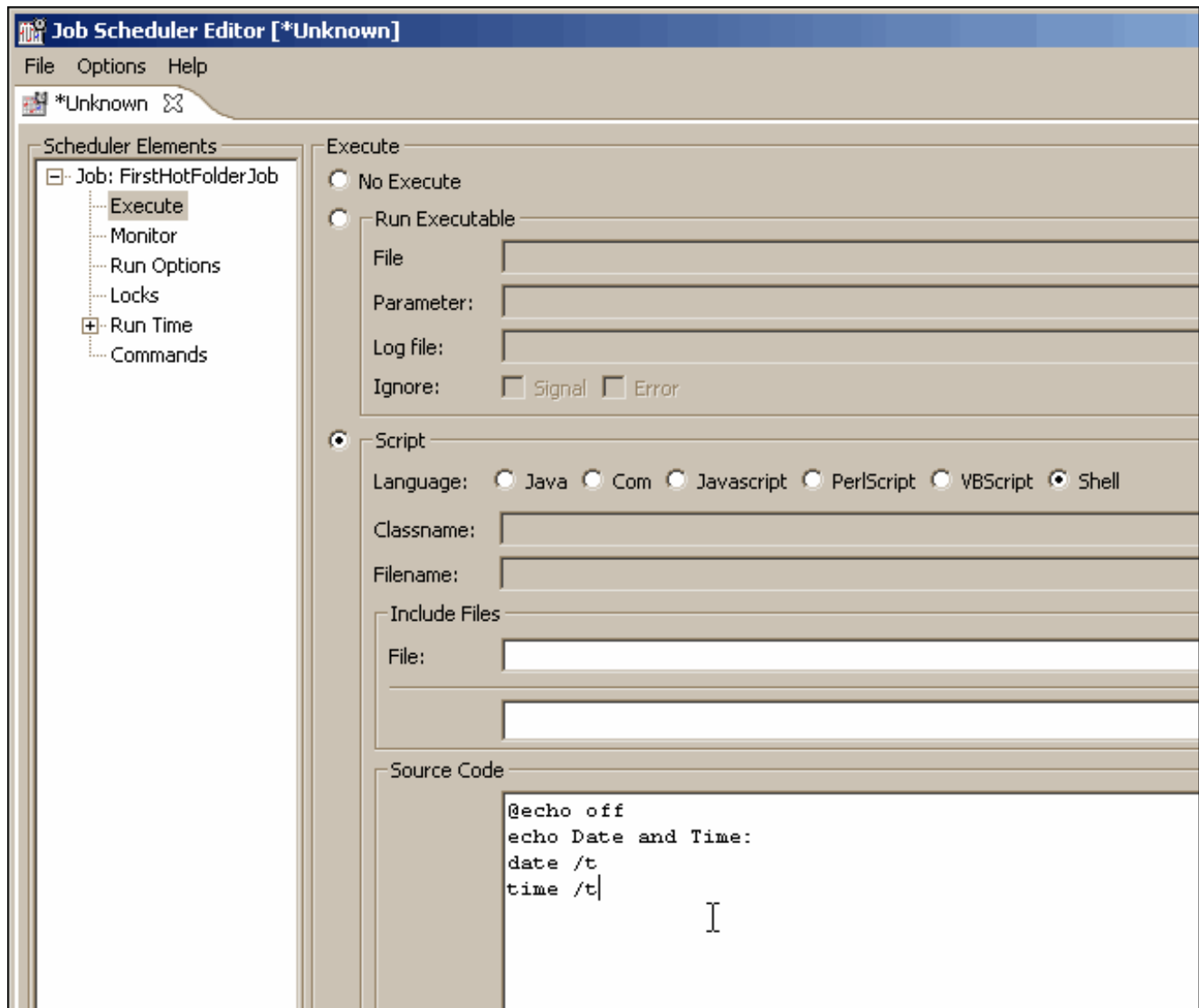
Wir wählen hier die Möglichkeit, den Code direkt in die Job-Konfiguration einzubetten. Wählen Sie dazu "Script" und als Sprache "Shell".

Geben Sie im Feld "Source Code" folgendes ein falls Sie unter Windows arbeiten:

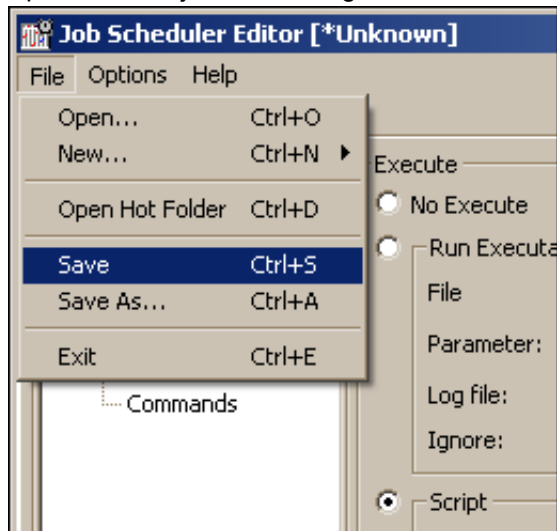
```
@echo off
echo Date and Time
date /t
time /t
```

Falls Sie unter Unix arbeiten geben Sie stattdessen ein:

```
#!/bin/sh
echo Date and Time
date
```



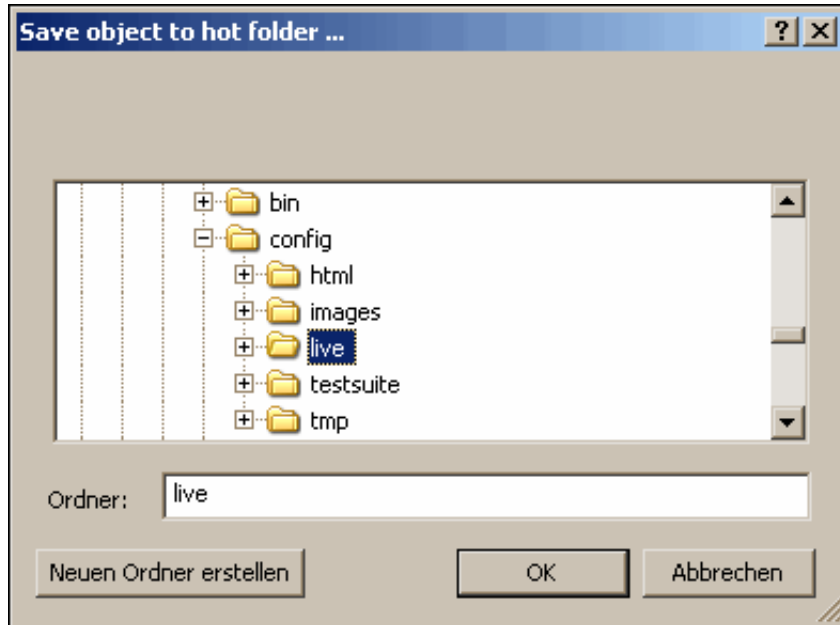
Speichern Sie jetzt die Konfiguration, indem sie auf **File->Save** klicken.



Es öffnet sich ein Dialog zur Angabe des "Hot Folders".

Per default ist das das Verzeichnis [scheduler install path]/config/live.

Wählen Sie also dieses Verzeichnis aus klicken Sie auf "OK".



Öffnen Sie jetzt die Weboberfläche, indem Sie im Browser die URL `http://host:port/` eingeben.

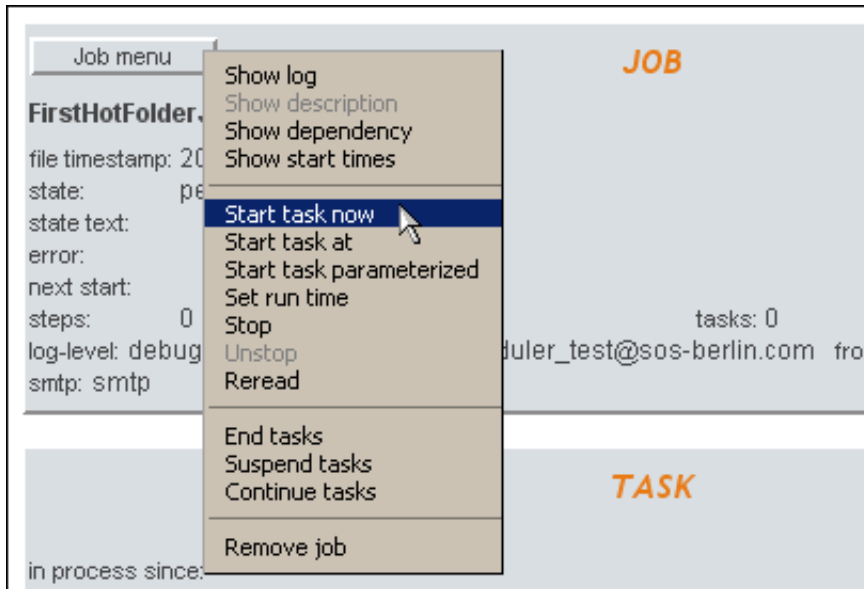
Für `host` und `port` sind die Werte Ihrer Job Scheduler Installation zu wählen.

Unter "Jobs" sehen Sie jetzt Ihren Job "FirstHotFolderJob".



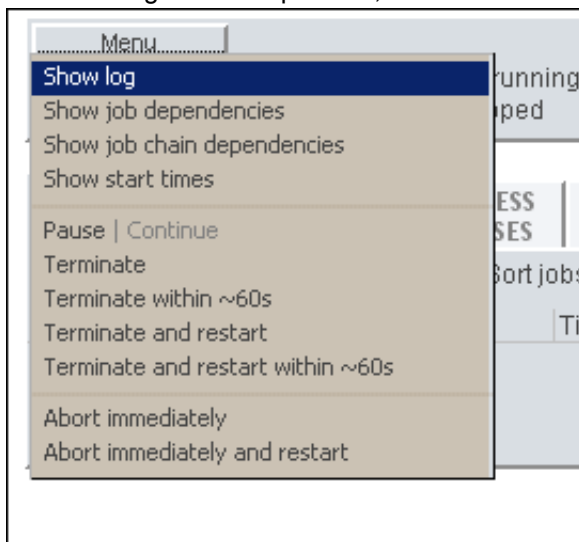
Klicken Sie auf den Job, um im rechten Frame des Browsers ins Job Menu zu gelangen.

Starten sie jetzt eine Instanz des Jobs, indem Sie auf Job menu->Start task now klicken.



Jetzt startet die Task.

Um die Ausgabe zu inspizieren, öffnen Sie das Log, indem Sie auf Menu->Show log klicken.



Jetzt öffnet sich ein neues Browser-Fenster mit der Log-Ausgabe des Schedulers.

Hier sehen Sie unter anderem die Ausgabe der Task.

```
[info] (Job FirstHotFolderJob) SCHEDULER-919 Task 2594397 enqueued
[info] (Job FirstHotFolderJob) SCHEDULER-930 Task 2594397 started - cause: queue_at
[info] (Task FirstHotFolderJob:2594397) SCHEDULER-918 state=starting (at=2007-12-03 21:16:51.968)
[info] (Task FirstHotFolderJob:2594397) SCHEDULER-987 Starting process: "C:\WINDOWS\TEMP\sos42C17.cmd"
[info] (Task FirstHotFolderJob:2594397) SCHEDULER-915 Process event
[info] (Task FirstHotFolderJob:2594397) stdout:
[info] (Task FirstHotFolderJob:2594397) Date and Time:
[info] (Task FirstHotFolderJob:2594397) 03.12.2007
[info] (Task FirstHotFolderJob:2594397) 21:16
[info] (Task FirstHotFolderJob:2594397) SCHEDULER-918 state=closed
```

4 Einen einfachen Job einrichten

Diese Anleitung beschreibt das Vorgehen zum Einrichten einer ausführbaren Datei als Job im Job Scheduler, nachdem der Job Scheduler erfolgreich installiert und eingerichtet wurde, sowohl unter Windows als auch unter Unix.

Ein neuer Job wird normalerweise in der Konfigurationsdatei `[scheduler install path]/config/scheduler.xml` eingetragen. Damit dieser neue Job vom Job Scheduler erkannt und verwaltet werden kann, ist ein Neustart des Job Schedulers erforderlich, dadurch wird die Konfigurationsdatei `scheduler.xml` neu eingelesen.

4.1 Voraussetzungen

- Der Job Scheduler wurde erfolgreich installiert.
- Sie habe eine ausführbare Datei vorbereitet, z.B. ein Kommandozeilen-Skript unter Windows oder ein Shell-Skript unter Unix. Diese Datei haben Sie im Verzeichnis `[scheduler install path]/jobs` abgelegt.

Beispiel Skripte:

Die folgenden beiden Skripte geben unter dem jeweiligen Betriebssystem auf der Kommandozeile die Überschrift "Datums- und Zeitausgabe:" und in der nächsten Zeile das aktuelle Datum und die aktuelle Zeit aus.

Windows: Inhalt des cmd-Skripts `my_first_job.cmd`:

```
@echo off
echo Datums- und Zeitausgabe:
date /t
time /t
```

Unix: Inhalt des shell-Skripts `my_first_job.sh`:

```
#!/bin/sh
echo Datums- und Zeitausgabe:
date
```

Bedenken Sie bei dem shell-Skript, dass der User des Job Schedulers die erforderlichen Rechte zur Ausführung des Skripts besitzen muss und das executable-Flag gesetzt ist.

4.2 Vorgehensweise

Öffnen Sie die Konfigurationsdatei `[scheduler install path]/config/scheduler.xml`.

Falls in der Datei der Tag `<jobs>` noch nicht existiert, fügen Sie bitte vor dem schließenden Tag `</config>` das Tag-Paar

```
<jobs>
</jobs>
```

ein.

Fügen Sie die folgenden Zeilen des `<job>`-Tags unter `<jobs>` ein, um das vorbereitete Skript als Job einzubinden.

Beispiel für Windows:

```
<jobs>

  <job name="my_first_job">
    <script language="shell">
      <include file="jobs\my_first_job.cmd"/>
    </script>
    <run_time repeat="10"/>
  </job>

</jobs>
```

Im Beispiel wird das Skript `jobs\my_first_job.cmd` als Job unter Windows eingebunden. Unter Unix wäre dementsprechend das Shell-Skript `jobs/my_first_job.sh` zu wählen.

`<run_time repeat="10"/>` bewirkt, dass der Job mit Start des Job Schedulers das erste mal gestartet wird. Der Job wird immer wieder gestartet, wobei zwischen Ende des einen und Start des nächsten Jobs einer Zeitdifferenz von 10 Sekunden besteht.

Speichern Sie die Änderungen in der Konfigurationsdatei `scheduler.xml` und starten Sie den Job Scheduler neu (die Konfigurationsdatei wird neu eingelesen).

Je nach verwendeter Hardware kann es unterschiedlich lange dauern, bis der Job Scheduler neu gestartet ist.

Öffnen Sie den Job Scheduler eigenen Web Server im Browser mit der URL

`http://[host]:[port]/`

Im Browser sehen Sie die Liste aller eingerichteten Jobs, unter anderem den soeben eingerichteten Job "my_first_job".

- Klicken Sie auf den Job "my_first_job".
- Das Fenster teilt sich in die Liste der Jobs und eine Übersicht des ausgewählten Jobs.
- Klicken Sie in der Übersicht des ausgewählten Jobs auf den Karteireiter "Task history".
- Sie sehen eine Auflistung der Start- und Endzeiten der bisherigen Durchläufe des Jobs. Die Differenz zwischen Endzeit eines Durchlaufs des Jobs und Startzeit des nächsten Durchlaufs beträgt 10 Sekunden, wie über `<run_time>` eingestellt wurde.

Herzlichen Glückwunsch!

Sie haben einen Job erfolgreich eingerichtet.

5 Einen einfachen Managed Job einrichten

Diese Anleitung beschreibt das Vorgehen zum Einrichten einer ausführbaren Datei als Managed Job im Job Scheduler, nachdem der Job Scheduler erfolgreich installiert und eingerichtet wurde, sowohl unter Windows als auch unter Unix.

Ein neuer Job wird normalerweise in der Konfigurationsdatei `[scheduler install path]/config/scheduler.xml` eingetragen. Damit dieser neue Job vom Job Scheduler erkannt und verwaltet werden kann, ist ein Neustart des Job Schedulers erforderlich, dadurch wird die Konfigurationsdatei `scheduler.xml` neu eingelesen.

Ein Managed Job unterscheidet sich von einem "normalen" Job u.a. dadurch, dass er über die Web-Oberfläche einzurichten ist - ein Anpassen von `scheduler.xml` ist nicht nötig - und die Daten des Jobs in einer Datenbank gespeichert werden.

5.1 Voraussetzungen

- Der Job Scheduler wurde erfolgreich mit den Paketen **Database Support**, **Web Interface** und **Managed Jobs** installiert. Falls Sie zwar den Job Scheduler aber ein oder mehrere benötigte Pakete nicht installiert haben, dann können Sie die Pakete auch nachträglich installieren. Siehe dazu die Dokumentation zur Installation und Konfiguration des Job Schedulers.
- Die Datenbank kann vom Job Scheduler angesprochen werden. Bei MySQL und SQL Server muss dem Job Scheduler ein zur verwendeten Datenbankversion kompatibler JDBC-Treiber bereitgestellt werden. Es ist uns nicht gestattet diese Treiber mitzuliefern, daher downloaden Sie sich bitte den entsprechenden Treiber
 - für MySQL von <http://www.mysql.com>
 - für SQL Server von <http://www.microsoft.com/downloads/details.aspx?FamilyID=e22bc83b-32ff-4474-a44a-22b6ae2c4e17&DisplayLang=en>
- Die Web-Oberfläche kann mit einem Browser geöffnet werden (der Web Server wurde entsprechend konfiguriert).
Beispiel für den Apache Web Server:
Angenommen, das Installationsverzeichnis des Job Schedulers heißt `my_scheduler/`.
In der Datei `httpd.conf` im Verzeichnis `conf/` des Apache Web Servers sollten die Aliase

```
Alias /my_scheduler/logs/      "C:/my_scheduler/logs/"
Alias /my_scheduler/          "C:/my_scheduler/web/"
```

für Windows, bzw.

```
Alias /my_scheduler/logs/     /home/[user]/my_scheduler/logs/
Alias /my_scheduler/         /home/[user]/my_scheduler/web/
```

für Unix eingetragen sein.

Nach einem Neustart des Apaches Web Servers läßt sich die Web-Oberfläche des Job Schedulers im Browser mit der URL `http://[hostname]/my_scheduler/index.htm` öffnen.

- Sie habe eine ausführbare Datei vorbereitet, z.B. ein Kommandozeilen-Skript unter Windows oder ein Shell-Skript unter Unix. Diese Datei haben Sie im Verzeichnis `[scheduler install path]/jobs` abgelegt.

Beispiel Skripte:

Die folgenden beiden Skripte geben unter dem jeweiligen Betriebssystem auf der Kommandozeile die Überschrift "Datums- und Zeitausgabe:" und in der nächsten Zeile das aktuelle Datum und die aktuelle Zeit aus.

Windows: Inhalt des cmd-Skripts `my_first_job.cmd`:

```
@echo off
echo Datums- und Zeitausgabe:
date /t
time /t
```

Unix: Inhalt des shell-Skripts `my_first_job.sh`:

```
#!/bin/sh
echo Datums- und Zeitausgabe:
date
```

Bedenken Sie bei dem shell-Skript, dass der User des Job Schedulers die erforderlichen Rechte zur Ausführung des Skripts besitzen muss und das executable-Flag gesetzt ist.

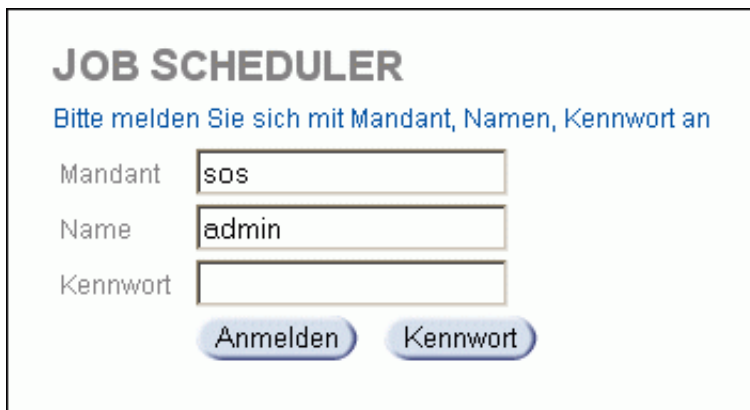
5.2 Vorgehensweise

Öffnen Sie die Web-Oberfläche des Job Schedulers in einem Browser mit der URL `http://[hostname]/my_scheduler/index.htm`

Es erscheint eine Login-Seite.

Geben Sie in das Formular bei Mandant `sos` und bei Name `admin` ein und lassen Sie das Kennwort frei.

Klicken Sie jetzt auf den Button "Anmelden".



JOB SCHEDULER

Bitte melden Sie sich mit Mandant, Namen, Kennwort an

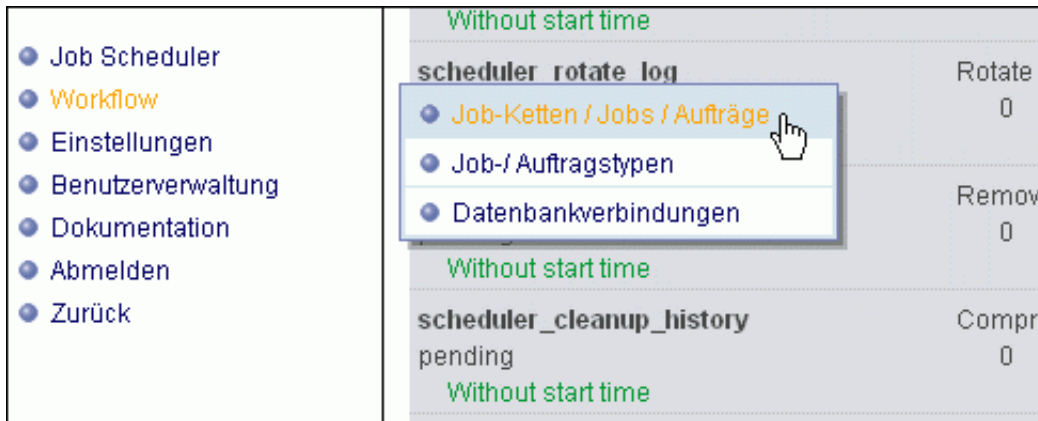
Mandant

Name

Kennwort

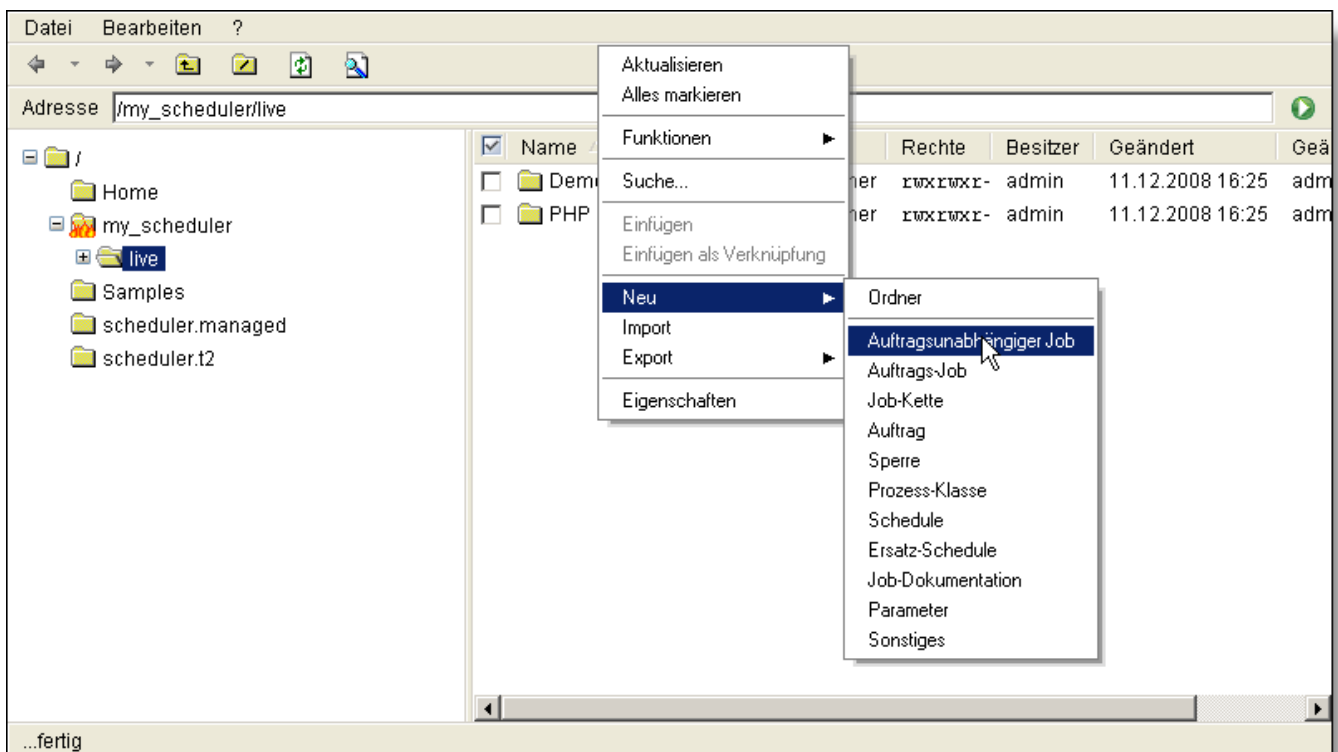
Sie sind jetzt angemeldet. Es erscheint die Hauptseite mit einem Auswahlnenü auf der linken Seite und einer Liste verschiedener Jobs auf der rechten Seite.

Klicken Sie im Auswahlnenü auf "Workflow"-> "Job-Ketten / Jobs / Aufträge".

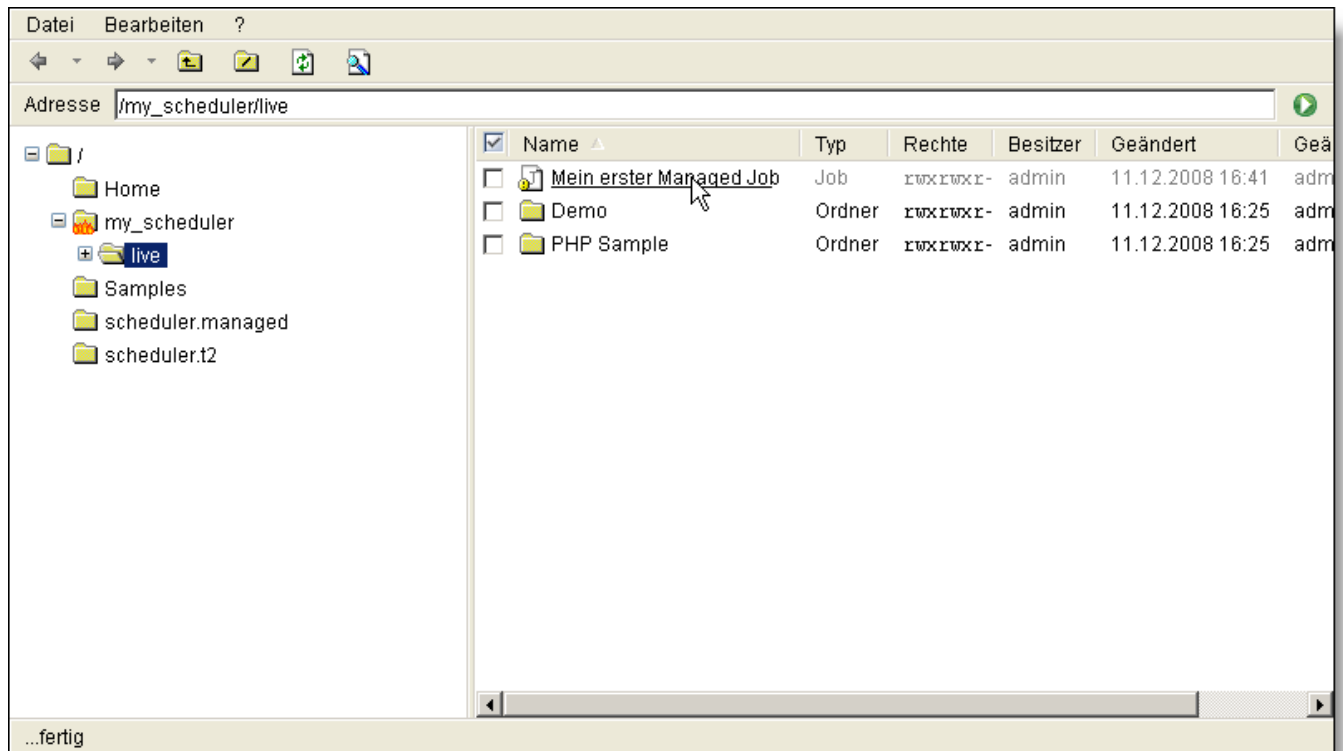


Auf der rechten Seite erscheint der Job Scheduler Explorer.

Wechseln Sie in das `live`-Verzeichnis ihres `Job Schedulers`. Legen Sie jetzt einen neuen Auftragsunabhängigen Job an, z.B. indem Sie im rechten Frame des Explorers im Kontextmenu "Neu"->"Auftragsunabhängiger Job" wählen



Nennen Sie den Job "Mein erster Managed Job".



Starten Sie jetzt den Job Editor, indem Sie im Explorer auf Ihren eben erstellten Job klicken.

Füllen Sie die Editormaske wie abgebildet aus:

- Geben Sie einen Titel für den Job an
- Wählen Sie als letztes bei Job-Konfiguration den Job-Typ "Executable File" und klicken Sie auf das Symbol "übernehmen":



Auftragsunabhängiger Job : Mein erster Managed Job

Job-Konfiguration

Job-Typ: Executable File DB-Verbindung: -

job
run_time

Title: Managed Jobs Demo

Process Class: []

Tasks: [] numeric

Min Tasks: [] numeric

Timeout: [] seconds | hh:mm:ss

Priority: -

Temporary: -

Java Options: []

Visible: -

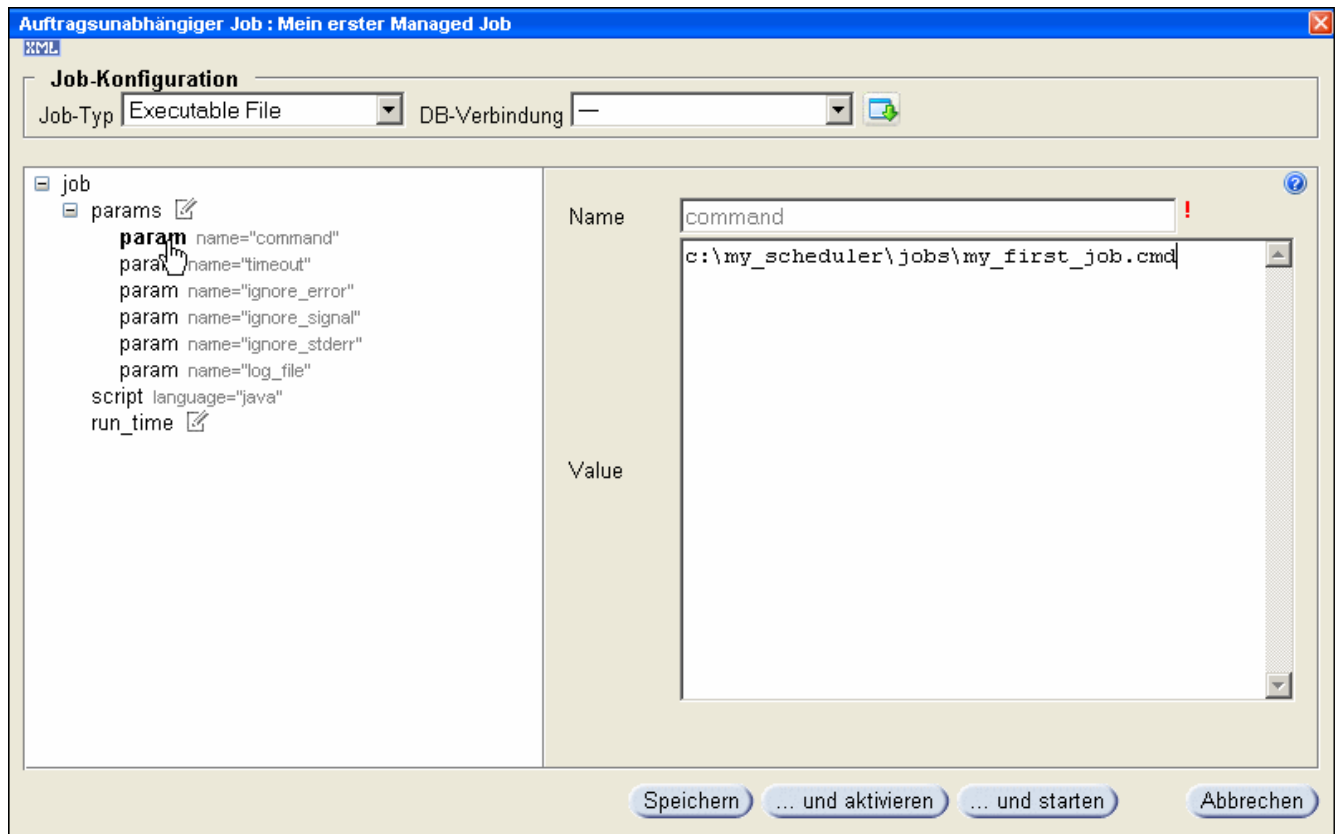
Ignore Signals: []

Stop On Error: -

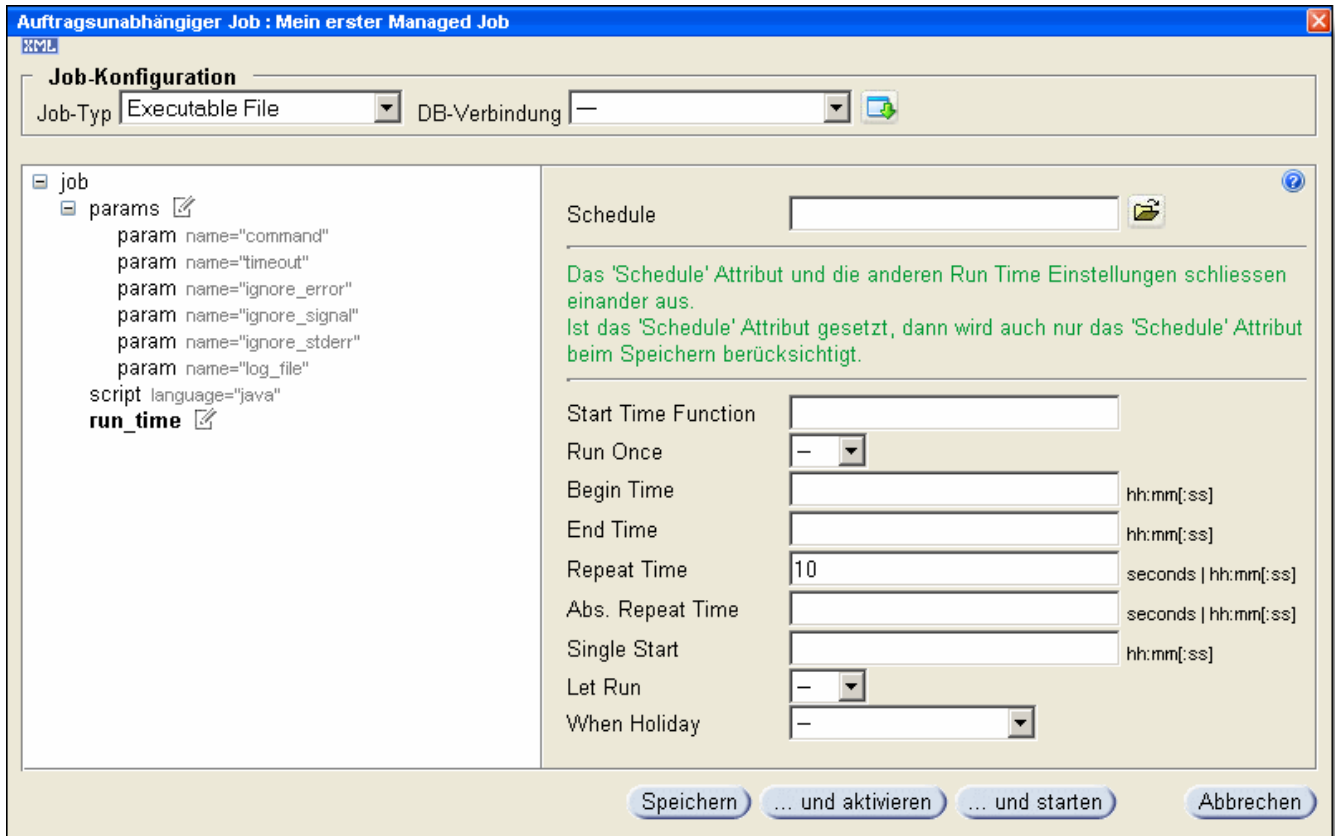
Speichern ... und aktivieren ... und starten Abbrechen

Der Job übernimmt jetzt die Parametermaske von "Executable File"

Tragen Sie bei dem Parameter `command` als Wert den Pfad des vorbereiteten Skripts ein.

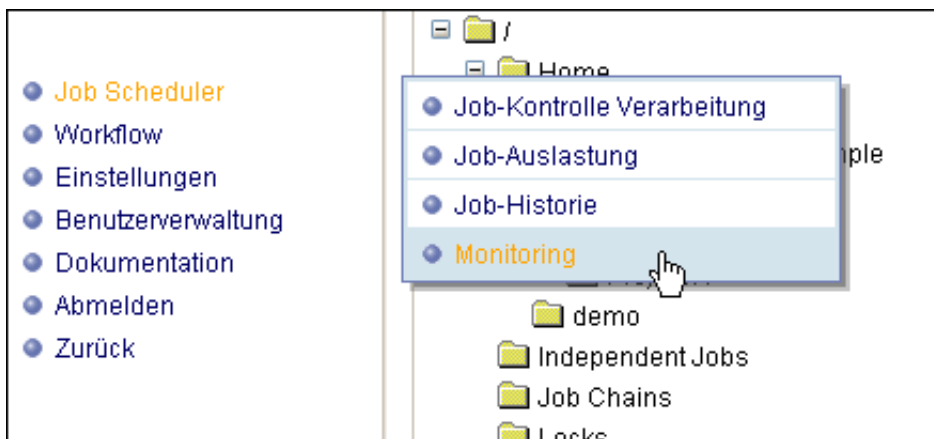


Stellen Sie nun die Laufzeit des Jobs über das Element `run_time` ein.
Tragen Sie bei dem Attribut `Repeat Time` 10 Sekunden ein.
Klicken Sie jetzt auf den Button zum Speichern und aktivieren.



Da für die Laufzeit des Job ein Repeat Time-Intervall eingestellt wurde, startet der Job Scheduler sofort nach dem Bestätigen den Job. Der Job wird immer wieder gestartet, wobei zwischen Ende der einen und Start der nächsten Task einer Zeitdifferenz von 10 Sekunden besteht.

Klicken Sie jetzt im Auswahlnenü auf "Job Scheduler"-> "Monitoring".



Es erscheint die Liste der Jobs. In der Liste ist jetzt auch der Job "Mein erster Managed Job" vertreten. Klicken in der Liste auf den Job "Mein erster Managed Job".

Die rechte Seite des Browsers-Fensters teilt sich in die Liste aller Jobs und eine Übersicht des grade ausgewählten Jobs.

Klicken Sie in der Übersicht des ausgewählten Jobs auf den Karteireiter "Task history".

Sie sehen eine Auflistung der Start- und Endzeiten der bisherigen Durchläufe des Jobs.

Die Differenz zwischen Endzeit eines Durchlaufs des Jobs und Startzeit des nächsten Durchlaufs beträgt 10 Sekunden, wie im Startintervall eingestellt wurde.

Herzlichen Glückwunsch!
Sie haben einen Managed Job erfolgreich eingerichtet.

6 Anwendungsfälle für Job-Ketten

6.1 Jobs in Reihe starten mit automatischer Fehlerbehandlung

- **Eigenschaften**
 - Ein Job wird erst ausgeführt, wenn sein Vorgänger erfolgreich abgeschlossen ist
 - Im Fehlerfall wird durch das Job-Attribut `stop_on_error="no"` für den einen Auftrag die Job-Kette im Fehlerzustand beendet.
 - Es ist kein Wiederanlaufen bzw. Fortsetzen des Auftrags möglich
 - Verhalten bei Neustart des Job Schedulers: gar keines, Auftrag steht in der Historie
- **Fehlerbehandlung**
 - automatisch: Auftrag wird beendet
 - eigene Fehlerbehandlungsjobs
 - Es wird automatisch eine Fehlermail verschickt

- **Einsatzgebiet**

Diese Art der Job-Konfiguration wird verwendet, wenn Fehler einzelner Aufträge zugelassen und automatisch gehandhabt werden sollen, da andere Aufträge für dieselbe Job-Kette höchst wahrscheinlich erfolgreich verarbeitet werden, selbst wenn ein einzelner Auftrag einen Fehler erzeugt. Es wird davon ausgegangen, dass Aufträge wiederholt erzeugt werden können, entweder manuell oder durch eine externe Applikation. Typischerweise werden hier Aufträge ohne Startzeiten verwendet.

- **Beispiel**

```
<job name="job1" order="yes" stop_on_error="no">
  ...
</job>

<job name="job2" order="yes" stop_on_error="no">
  ...
</job>

<job_chain name="Chain_1">
  <job_chain_node state="1"
    job="job1"
    next_state="2"
    error_state="err"/>

  <job_chain_node state="2"
    job="job2"
    next_state="end"
    error_state="err"/>

  <job_chain_node state="end"/>
  <job_chain_node state="err"/>
</job_chain>
```

Mit Hilfe der Job-Kette `Chain_1` ist gewährleistet, dass der Job `job2` erst nach dem Durchlauf von Job `job1` startet. Falls in Job `job1` oder Job `job2` ein Fehler auftritt, wird der Fehlerzustand `err` erreicht.

6.2 Jobs in Reihe starten mit manueller Fehlerbehandlung

- **Eigenschaften**

- Ein Job wird erst ausgeführt, wenn sein Vorgänger erfolgreich abgeschlossen ist.
- Im Fehlerfall wird durch das Job-Attribut `stop_on_error="yes"` der Auftrag vor dem Job in die Auftragswarteschlange eingereiht. Alle weiteren Aufträge werden vor diesem Job ebenfalls in die Auftragswarteschlange eingereiht. Das heißt, dass die Aufträge nur bis vor diesen Job laufen und die Job-Kette an dieser Stelle gestoppt ist.
- Die Unterbrechung der Job-Kette durch einen gestoppten Job kann nur manuell aufgehoben werden.
- Verhalten bei Neustart des Job Schedulers:

Wenn eine Datenbank verwendet wird und das Job-Kettenattribut `orders_recoverable="yes"` gesetzt ist (default), dann werden die Aufträge der Auftragswarteschlange aus der Datenbank geholt und laufen wieder an. Wenn der Job unverändert geblieben ist, dann wird beim ersten Auftrag wieder ein Fehler auftreten und der job stoppen. Wenn kein Datenbank verwendet wird oder das Job-Kettenattribut `orders_recoverable="no"` gesetzt ist, ist die Job-Kette wieder bereit für neue Aufträge. Die alten Aufträge aus der Auftragswarteschlange sind verloren.

- **Fehlerbehandlung**

- Job wird automatisch gestoppt
- Es wird automatisch eine Fehlermail verschickt
- manuelles Eingreifen des Administrators ist erforderlich:

Der Administrator kann nach dem Beheben der Fehlerursache den Job mit "Unstop" wieder weiterlaufen lassen. Alle wartenden Aufträge einschließlich des Auftrags bei dem der Fehler aufgetreten ist laufen in der Job-Kette weiter.

- **Einsatzgebiet**

Diese Art der Job-Konfiguration wird verwendet, wenn davon ausgegangen wird, dass die Ursache von Fehlern beim Job liegt und nicht bei den einzelnen Aufträgen. Man nimmt an, dass beim Auftreten eines Fehlers auch bei allen anderen Aufträgen in diesem Job Fehler entstehen würden. Durch Verwendung einer Datenbank, können die Aufträge bewahrt werden. Man verwendet diese Art von Konfiguration, wenn Aufträge nicht wiederholt erzeugt werden können. Typischerweise werden hier Aufträge ohne Startzeiten verwendet.

- **Beispiel**

```
<job name="job1" order="yes" stop_on_error="yes">
  ...
</job>

<job name="job2" order="yes" stop_on_error="yes">
  ...
</job>

<job_chain name="Chain_2" orders_recoverable="yes">
  <job_chain_node state="1"
    job="job1"
    next_state="2"/>

  <job_chain_node state="2"
    job="job2"
    next_state="end"/>

  <job_chain_node state="end"/>
</job_chain>
```

Mit Hilfe der Job-Kette `Chain_2` ist gewährleistet, dass der job `job2` erst nach dem Durchlauf von Job `job1` startet. Falls in Job `job1` oder Job `job2` ein Fehler auftritt, wird der betreffende Job angehalten. Wenn in

einem Job `stop_on_error="yes"` steht, ist eine Angabe des Fehlerzustands `error_state` in dem entsprechenden Job-Kettenknoten überflüssig, da ein Fehlerzustand des Auftrags nie erreicht wird. Ein Auftrag wird in demselben Job solange wiederholt, bis dieser erfolgreich durchlaufen wurde. Danach geht der Auftrag in den positiven Folgezustand `next_state` über.

6.3 Jobs in Reihe starten mit automatischer Wiederholung im Fehlerfall

- **Eigenschaften**
 - Ein Job wird erst ausgeführt, wenn sein Vorgänger erfolgreich abgeschlossen ist.
 - Im Fehlerfall wird durch das Job-Attribut `stop_on_error="no"` und das Job-Kettenattribut `on_error="setback"` der Auftrag zurückgestellt und nach einer bestimmten Verzögerung in dem Job, in dem der Fehler aufgetreten ist wiederholt.
 - Wenn bei einem erneuten Job-Durchlauf kein Fehler auftritt, läuft der Auftrag in der Job-Kette normal weiter.
 - Wenn nach einer eingestellten Anzahl von erlaubten Wiederholungen immer noch ein Fehler auftritt, wird für diesen Auftrag die Job-Kette im Fehlerzustand beendet.
 - Andere Aufträge bleiben von dem Fehlerverhalten des einzelnen Auftrags unbeeinflusst.
 - Verhalten bei Neustart des Job Schedulers: gar keines, Auftrag steht in der Historie

- **Fehlerbehandlung**
 - Es wird automatisch eine Fehlermail verschickt
 - automatisch: Auftrag wird n-mal wiederholt
 - eigene Fehlerbehandlungsjobs

- **Einsatzgebiet**

Diese Art der Job-Konfiguration wird verwendet, wenn Fehler einzelner Aufträge zugelassen und automatisch gehandhabt werden sollen, da davon ausgegangen wird, dass andere Aufträge für dieselbe Job-Kette erfolgreich ablaufen können, selbst wenn ein einzelner Auftrag einen Fehler erzeugt. Man geht davon aus, dass der Auftrag mit hoher Wahrscheinlichkeit den Job spätestens nach der n-ten Wiederholung erfolgreich durchläuft. Die Ursache des Fehlers löst sich von selbst, wenn nur genug Zeit vergangen ist. Aufträge können wiederholt erzeugt werden, entweder manuell oder durch eine externe Applikation, falls das Problem durch die Wiederholungen nicht gelöst werden konnte. Typischerweise werden hier Aufträge ohne Startzeiten verwendet.

- **Beispiel**

```
<job name="job1" order="yes" stop_on_error="no">
```

```
...
```

```
<delay_order_after_setback
  setback_count="2"
  delay="5"/>
```

```
<delay_order_after_setback
  setback_count="4"
  delay="10"/>
```

```
<delay_order_after_setback
  setback_count="5"
  is_maximum="yes"/>
```

```
</job>
```

```
<job name="job2" order="yes" stop_on_error="no">
```

```
...
```

```

</job>

<job_chain name="Chain_3">
  <job_chain_node state="1"
    job="job1"
    next_state="2"
    error_state="err"
    on_error="setback"/>

  <job_chain_node state="2"
    job="job2"
    next_state="end"
    error_state="err"/>

  <job_chain_node state="end"/>
  <job_chain_node state="err"/>
</job_chain>

```

Falls in Job `job1` ein Fehler auftritt, wird der Auftrag beim ersten Mal zurückgesetzt und durchläuft den Job sofort noch einmal. Tritt dann wieder ein Fehler auf (zum 2. Mal), wird wegen `setback_count="2"` der Auftrag 5 Sekunden verzögert bis er den Job noch einmal durchläuft. Beim 3. Fehler wird der Auftrag ebenfalls um 5 Sekunden verzögert. Beim 4. Fehler wird der Auftrag um 10 Sekunden verzögert bis er den Job erneut zu durchlaufen versucht. Beim 5. Fehler wird der Auftrag ebenfalls um 10 Sekunden verzögert. Wenn danach noch ein Fehler auftritt, geht der Auftrag in den Fehlerzustand `err` über. Falls in Job `job2` ein Fehler auftritt, wird direkt der Fehlerzustand `err` erreicht. Wenn in einem Job-Kettenknoten `on_error="setback"` verwendet wird, muss im zugehörigen Job `<delay_order_after_setback>` konfiguriert sein.

6.4 Jobs in Reihe starten mit manueller Wiederholung im Fehlerfall

- **Eigenschaften**
 - Ein Job wird erst ausgeführt, wenn sein Vorgänger erfolgreich abgeschlossen ist.
 - Im Fehlerfall wird durch das Job-Attribut `stop_on_error="no"` und das Job-Kettenattribut `on_error="suspend"` der Auftrag zurückgestellt und wartet vor dem Job ausserhalb der Auftragswarteschlange.
 - Der Auftrag kann nur manuell reaktiviert werden.
 - Wenn bei einem erneuten Job-Durchlauf kein Fehler auftritt, läuft der Auftrag in der Job-Kette normal weiter.
 - Andere Aufträge bleiben von dem Fehlerverhalten des einzelnen Auftrags unbeeinflusst.
 - Verhalten bei Neustart des Job Schedulers:
Wenn eine Datenbank verwendet wird und das Job-Kettenattribut `orders_recoverable="yes"` gesetzt ist (default), dann werden die suspendierten Aufträge aus der Datenbank geholt und warten vor dem Job auf ihre Reaktivierung. Wenn keine Datenbank verwendet wird oder das Job-Kettenattribut `orders_recoverable="no"` gesetzt ist, sind die ursprünglich suspendierten Aufträge verloren.
- **Fehlerbehandlung**
 - Es wird automatisch eine Fehlermail verschickt
 - Der Auftrag wird vor dem Job angehalten
 - manuelles Eingreifen des Administrators ist erforderlich:
Der Administrator kann nach dem Beheben der Fehlerursache den Auftrag wieder anlaufen lassen.
- **Einsatzgebiet**

Diese Art der Job-Konfiguration wird verwendet, wenn Fehler einzelner Aufträge manuell gehandhabt werden sollen. Es wird davon ausgegangen, dass andere Aufträge die Job-Kette erfolgreich durchlaufen können,

selbst wenn ein einzelner Auftrag einen Fehler erzeugt. Falls ein Fehler auftritt, stellt eine automatische Wiederholung des Auftrags keine erfolgsversprechende Problemlösung dar. Man verwendet diese Art von Konfiguration, wenn Aufträge nicht wiederholt erzeugt werden können und keine Weiterverarbeitung des Auftrags nach einem Fehler möglich ist. Typischerweise werden hier Aufträge ohne Startzeiten verwendet.

• **Beispiel**

```

<job name="job1" order="yes" stop_on_error="no">
  ...
</job>

<job name="job2" order="yes" stop_on_error="no">
  ...
</job>

<job_chain name="Chain_4" orders_recoverable="yes">
  <job_chain_node state="1"
    job="job1"
    next_state="2"
    on_error="suspend"/>

  <job_chain_node state="2"
    job="job2"
    next_state="end"
    on_error="suspend"/>

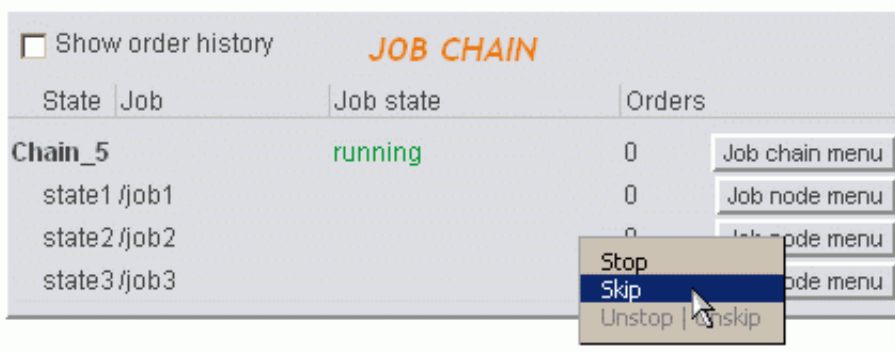
  <job_chain_node state="end"/>
</job_chain>

```

Falls in Job `job1` oder Job `job2` ein Fehler auftritt, wird der Auftrag vor dem jeweiligen Job ausserhalb der Auftragswarteschlange angehalten. Aufträge können nicht in einen Folgezustand `error_state` gelangen.

6.5 Manuelles Überspringen von Jobs in einer Job-Kette

Der Administrator kann bei einem Job-Kettenknoten manuell einstellen, dass er bei der Auftragsverarbeitung übersprungen werden soll ("Skip"). Dadurch wird ein die Job-Kette durchlaufender Auftrag an den nachfolgenden Knoten der Job-Kette und dessen Job weitergeleitet. Mit "Unskip" kann der Knoten wieder in die Job-Kette eingehängt werden.



JOB CHAIN			
State	Job	Job state	Orders
Chain_5		running	0
	state1 /job1		0
	state2 /job2	skipped	
	state3 /job3		

6.6 Manuelles Anhalten von Jobs in einer Job-Kette

Der Administrator kann einen Job jederzeit manuell stoppen ("Stop"). In diesem Fall warten ankommende Aufträge solange in der Auftragswarteschlange bis der Job wieder gestartet wird und laufen dann in der Job-Kette weiter. Bei dem Vorgang wird kein Fehler erzeugt und somit keine Fehlermail verschickt. Mit "Unstop" startet der Administrator den Job wieder.

JOB CHAIN			
State	Job	Job state	Orders
Chain_5		running	0
	state1 /job1		0
	state2 /job2		
	state3 /job3		

JOB CHAIN			
State	Job	Job state	Orders
Chain_5		running	0
	state1 /job1		0
	state2 /job2	stopped	
	state3 /job3		

6.7 Auf Verzeichnisänderungen reagieren

- Aufgabe**
 Es soll ein Mandantenfähiges System entstehen, das auf das Eintreffen von Dateien reagiert. Anhand jeder Datei kann der Mandant identifiziert werden. Jede Datei steht für einen Auftrag dieses Mandanten. Der selbe Programmablauf soll wiederholt für verschiedene Mandanten stattfinden. Falls es bei einem Auftrag Probleme geben sollte, soll das System andere Aufträge weiterhin verarbeiten.
- Lösung**

Hier bietet sich eine Verzeichnisüberwachung mittels Dateiaufträgen an.

Ein beliebiges Verzeichnis wird dazu als überwachtes Verzeichnis eingestellt. Wenn eine Datei in das Verzeichnis kopiert wird, triggert die Überwachung und es wird ein Auftrag bezüglich der Datei erzeugt.

- **Beispiel**

```
<job name="job_1" order="yes" stop_on_error="no">
```

```
...
```

```
</job>
```

```
<job name="job_2" order="yes" stop_on_error="no">
```

```
...
```

```
</job>
```

```
<job name="job_3" order="yes" stop_on_error="no">
```

```
...
```

```
</job>
```

```
<job_chain name="Chain_A">
```

```
  <file_order_source directory="path"/>
```

```
  <job_chain_node state="state_1" job="job_1" error_state="error"/>
```

```
  <job_chain_node state="state_2" job="job_2" error_state="error"/>
```

```
  <job_chain_node state="state_3" job="job_3" error_state="error"/>
```

```
  <file_order_sink state="ok" remove="yes"/>
```

```
  <file_order_sink state="error" move_to="errorpath"/>
```

```
</job_chain>
```

Wenn eine Datei in das Verzeichnis `path` kopiert wird, reagiert die Verzeichnisüberwachung und ein Auftrag wird erzeugt.

Dieser wird zunächst von Job `job_1`, dann von Job `job_2` und zuletzt von Job `job_3` verarbeitet.

Sollte während der Verarbeitung des Auftrags ein Fehler auftreten, wird der Auftrag aus der Job-Kette entfernt und die zugehörige Datei in das Verzeichnis `errorpath` verschoben. Wenn die Verarbeitung in allen drei Jobs erfolgreich war, wird die den Auftrag auslösende Datei gelöscht.

Die Job-Kette bleibt in jedem Fall für weitere Dateiaufträge zur Auftragsverarbeitung offen.

- **Anmerkung**

Es existiert auch die Möglichkeit eine Verzeichnisüberwachung für einen Standalone Job einzustellen.

Siehe `<start_when_directory_changed>`

Der Unterschied zur Verzeichnisüberwachung mit Dateiaufträgen besteht darin, dass pro Datei ein Dateiauftrag erzeugt wird.

Bei der Job-Verzeichnisüberwachung kann man nur das Verzeichnis als ganzes überwachen, d.h. der Job reagiert auf jede neue und jede gelöschte Datei. Allerdings muss der Job selbst herausfinden was für eine Änderung stattgefunden hat.

Der Standalone Job arbeitet also nur auf der Dateimenge des Verzeichnisses.

Für eine einfache Überwachung leistet dieses kompakte Verfahren jedoch gute Dienste.

Anhang A: Beispiel 1: Zeitgesteuertes Ausführen eines Shell-Skripts

In diesem Beispiel ist der Job `my_shell_script` wie folgt eingerichtet:

- werktags im Zeitraum 9-12h halbstündlicher automatischer Start
- Darüber hinaus kann der Job beliebig im Zeitraum 8:00-20:00 manuell gestartet werden.
- Der Job führt das Shell-Skript `./jobs/my_shell_script.sh` aus.
- Die Zeitsteuerung ist mit einer Verzeichnisüberwachung kombinierbar (siehe Beispiel 2).

In die XML-Konfigurationsdatei `./config/scheduler.xml` muss ein Job-Element wie folgt eingefügt werden:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<spooler>
  <config ...>
    <security ignore_unknown_hosts = "yes">
      <allowed_host host = "localhost" level = "all"/>
    </security>
    <process_classes>
      ...
    </process_classes>
    <jobs>
      <job name = "my_shell_script">
        <process file = "jobs/my_shell_script.sh"
          <!-- in Windows analog "jobs\my_shell_script.cmd" -->
          param = "">
        </process>
        <run_time begin = "08:00"
          end = "20:00">
          <weekdays>
            <day day="1">
              <period begin = "09:00" end = "12:00" repeat = "1800"/>
            </day>
            <day day="2">
              <period begin = "09:00" end = "12:00" repeat = "1800"/>
            </day>
            <day day="3">
              <period begin = "09:00" end = "12:00" repeat = "1800"/>
            </day>
            <day day="4">
              <period begin = "09:00" end = "12:00" repeat = "1800"/>
            </day>
            <day day="5">
              <period begin = "09:00" end = "12:00" repeat = "1800"/>
            </day>
          </weekdays>
        </run_time>
      </job>
    </jobs>
  </config>
</spooler>
```

Anhang B: Beispiel 2: Ausführen eines Shell-Skripts per Verzeichnis-Überwachung

In diesem Beispiel wird das Verzeichnis `./notification_dir` überwacht.

- Wann immer in diesem Verzeichnis eine Datei mit der Dateierweiterung `txt` hinzugefügt oder gelöscht wird, wird der Job `my_shell_script` vom Job Scheduler automatisch gestartet. Darüber hinaus kann der Job jederzeit manuell gestartet werden.
- Der Job führt das Shell-Skript `./jobs/my_shell_script.sh` aus.
- Die Verzeichnis-Überwachung ist auch mit einer Zeitsteuerung kombinierbar (siehe Beispiel 1).

In die XML-Konfigurationsdatei `./config/scheduler.xml` muss ein Job-Element wie folgt eingefügt werden:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<spooler>
  <config ...>
    <security ignore_unknown_hosts = "yes">
      <allowed_host host = "localhost" level = "all"/>
    </security>
    <process_classes>
      ...
    </process_classes>
    <jobs>
      <job name = "my_shell_script">
        <process file = "jobs/my_shell_script.sh"
          <!-- in Windows analog "jobs\my_shell_script.cmd" -->
          param = "">
        </process>
        <start_when_directory_changed directory = "notification_dir"
          regex = "\.txt$"/>
        <run_time/>
      </job>
    </jobs>
  </config>
</spooler>
```

Anhang C: Beispiel 3: Ausführen eines PHP-Skripts

In diesem Beispiel kann der Job `my_php_script` lediglich manuell gestartet werden.

- Der Job führt das Command-Line-Interface von PHP aus, welches das PHP-Skript als Parameter übergeben bekommt.
- Wenn dem PHP-Skript Parameter übergeben werden sollen, so sind diese mit Leerzeichen getrennt dahinter anzugeben.
- Für die Zeitsteuerung und Verzeichnis-Überwachung siehe Beispiel 1 und Beispiel 2.

In die XML-Konfigurationsdatei `./config/scheduler.xml` muss ein Job-Element wie folgt eingefügt werden:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<spooler>
  <config ...>
    <security ignore_unknown_hosts = "yes">
      <allowed_host host = "localhost" level = "all"/>
    </security>
    <process_classes>
      ...
    </process_classes>
    <jobs>
      <job name = "my_php_script">
        <process file = "/usr/src/php-4.3.3/sapi/cli/php"
          <!-- in Windows meist "c:\php\cli\php.exe" -->
          param = "-f jobs/my_php_script.php">
        </process>
        <run_time/>
      </job>
    </jobs>
  </config>
</spooler>
```

Anhang D: Beispiel 4: Programme ausführen

In diesem Beispiel wird mittels zweier Jobs um 4:00 Uhr nachts der Windows-Dienst der MySQL-Datenbank neu gestartet.

- Der erste Job `service_stop` stoppt und der zweite Job `service_start` startet den Windows-Dienst. Hierzu wird das Programm `net.exe` des Windows Betriebssystems verwendet. Dieses liegt im Unterverzeichnis `system32` des Windows-Verzeichnisses. Für den Pfad kann die Umgebungsvariable `%windir%` verwendet werden. Umgebungsvariablen sind auch unter Windows mit einem `$` zu kennzeichnen.
- Der Dienstname der MySQL-Datenbank sei `MySQL`.
- Für dieses Beispiel wäre es besser ein entsprechendes Kommandozeilen-Skript einzubinden, damit Fehlerzustände abgefangen werden können.

In die XML-Konfigurationsdatei `./config/scheduler.xml` müssen zwei Job-Element wie folgt eingefügt werden:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<spooler>
  <config ...>
    <security ignore_unknown_hosts = "yes">
      <allowed_host host = "localhost" level = "all"/>
    </security>
    <process_classes>
      ...
    </process_classes>
    <jobs>
      <job name = "service_stop">
        <process file = "$windir\system32\net.exe"
          param = "stop MySQL">
        </process>
        <run_time>
          <period single_start = "04:00"/>
        </run_time>
      </job>
      <job name = "service_start">
        <process file = "$windir\system32\net.exe"
          param = "start MySQL">
        </process>
        <run_time>
          <period single_start = "04:01"/>
        </run_time>
      </job>
    </jobs>
  </config>
</spooler>
```

Glossar

Auftrag

Ein Auftrag aktiviert die Verarbeitung einer Job-Kette. Der Auftrag enthält Parameter für einen oder mehrere Jobs einer Job-Kette. Ein persistenter Auftrag enthält eine Startzeit bzw. ein Wiederholungsintervall, z.B. für täglich wiederholten Start. Ist der Auftrag nicht persistent, dann wird er nach Abschluß der Verarbeitung aus der Job-Kette entfernt.

Ein Auftrag durchläuft nacheinander die Jobs einer Job-Kette. Tritt ein Verarbeitungsfehler in einem Job auf, dann wird der Auftrag gestoppt und aus der Job-Kette entfernt. Jeder Job der Job-Kette hat Zugriff auf die Parameter eines Auftrags.

Job

Verarbeitungseinheit des Job Schedulers.

Programme und Skripte, die vom Job Scheduler ausgeführt werden sollen, müssen in Jobs eingebettet werden. Jobs können beliebige ausführbare Dateien starten oder Job-Skripte enthalten, die die Programmschnittstelle des Job Schedulers verwenden. Jobs können in mehreren Prozessen (tasks) ablaufen, wenn dies zur Skalierung der Leistung gewünscht ist.

Job-Kette

Eine Reihe von Jobs, die nacheinander Aufträge verarbeiten.

Der Job Scheduler startet die Jobs einer Job-Kette automatisch, wenn ein Auftrag eintrifft. Job-Ketten bieten die Möglichkeit mehrere Aufträge parallel zu verarbeiten, indem die Jobs in mehreren Instanzen gestartet werden.

Managed Jobs

Jobs, die in einer Datenbank verwaltet und automatisch an einen oder mehrere Job Scheduler verteilt werden. Für den Betrieb wird eine Datenbank vorausgesetzt.