Initiator: Generisches Python-Template für "Process"

Generisches Python-Template für Initiatoren

Ein Initiator erlaubt als das aktive Verändern des Zustand einer Ressource (siehe hier für erlaubte Zustände), an die er gebunden ist. Diese auf dieser Seite z ufindenen Templates schreiben ohne besondere Logik verschiedene Zustände einer Ressource für einen Nutzer.

Sie haben damit nur beschränkt reellen Nutzen, zeigen jedoch, wie ein Initiator (auch) zu bedienen ist.

Initiator Beispiel für BFFH-Initialisierung (einmalig beim Start)

```
mkdir -p /opt/fabinfra/adapters/initiator_process_example/
vim init.once.py
```

```
#!/usr/bin/env python

import os
import sys
import time
import logging
LOG_FILE = os.path.join(os.path.dirname(__file__), 'init.once.log')
logging.basicConfig(
    filename=LOG_FILE,
    format='%(asctime)s %(levelname)-8s %(message)s',
    level=logging.DEBUG,
    datefmt='%Y-%m-%d %H:%M:%S')

try:
    # Statements, die per print() an stdout gegeben werden, werden in bffhd geschrieben

    # Versetze die Ressource (Maschine) in den Zustand "Reserved" durch den Nutzer "Admin"
    state = '{ "state": { "Reserved": { "id": "Admin" } } }'
```

```
    print(state)
    sys.stdout.flush()
    logging.debug(state)

    # wir brauchen wenigstens ein paar Sekunden nach dem letzten flush, damit alles an bffhd
übertragen wird
    # falls der Status nicht verändert wird: in Betracht ziehen die Sleep Time zu erhöhen
    time.sleep(3)

except Exception as e:
    logging.debug(e)
    sys.exit(1)

sys.exit(0)
```

```
chmod +x /opt/fabinfra/adapters/initiator_process_example/init.once.py
```

Initiator Beispiel in Dauerschleife

Dieses Sample geht in Dauerschleife alle verfügbaren Zustände durch und schreibt sie dem Nutzer `Admin` zu.

```
mkdir -p /opt/fabinfra/adapters/initiator_process_example/
vim init.loop.py
```

```
#!/usr/bin/env python

import os
import sys
import time
import logging
LOG_FILE = os.path.join(os.path.dirname(__file__), 'init.loop.log')
logging.basicConfig(
    filename=LOG_FILE,
    format='%(asctime)s %(levelname)-8s %(message)s',
    level=logging.DEBUG,
    datefmt='%Y-%m-%d %H:%M:%S')

# Init sleep
time.sleep(10)
```

```
while True:
    try:
        # Statements, die per print() an stdout gegeben werden, werden in bffhd geschrieben

        t_sleep = 3 #darf nicht 0 sein. Sonst wechseln die Zustände zu schnell durch und verursachen Fehler und ein gefülltes BFFH Log (journalctl)

        # Versetze die Ressource (Maschine) in den Zustand "Free"
        state = '{ "state": "Free" }'
        print(state)
        sys.stdout.flush()
        logging.debug(state)
        time.sleep(t_sleep)

        # Versetze die Ressource (Maschine) in den Zustand "Blocked" durch den Nutzer "Admin"
        state = '{ "state": { "InUse": { "id": "Admin" } } }'
        print(state)
        sys.stdout.flush()
        logging.debug(state)
        time.sleep(t_sleep)

        # Versetze die Ressource (Maschine) in den Zustand "Reserved" durch den Nutzer "Admin"
        state = '{ "state": { "Reserved": { "id": "Admin" } } }'
        print(state)
        sys.stdout.flush()
        logging.debug(state)
        time.sleep(t_sleep)

        # Versetze die Ressource (Maschine) in den Zustand "ToCheck" durch den Nutzer "Admin"
        state = '{ "state": { "ToCheck": { "id": "Admin" } } }'
        print(state)
        sys.stdout.flush()
        logging.debug(state)
        time.sleep(t_sleep)

        # Versetze die Ressource (Maschine) in den Zustand "Blocked" durch den Nutzer "Admin"
        state = '{ "state": { "Blocked": { "id": "Admin" } } }'
        print(state)
        sys.stdout.flush()
```

```
        logging.debug(state)
        time.sleep(t_sleep)

        # Versetze die Ressource (Maschine) in den Zustand "Disabled"
        state = '{ "state": "Disabled" }'
        print(state)
        sys.stdout.flush()
        logging.debug(state)
        time.sleep(t_sleep)

    except Exception as e:
        logging.debug(e)
        sys.exit(1)

sys.exit(0)
```

```
chmod +x /opt/fabinfra/adapters/initiator_process_example/init.loop.py
```

> **Achtung:** Der Initiator wird nicht automatisch neu von BFFH aufgerufen, wenn das Script scheitert (z.B. wegen Schreib- oder Lesefehlern oder sonstigen Exceptions). In diesem Fall muss der BFFH Server neugestartet werden oder mehr Programmlogik in den Initiator eingebaut werden, damit sich das Script selbst neu startet!

bffh.dhall Snippet

Wir binden den Inititator entsprechend in unsere BFFH Konfiguration ein.

```
    , initiators =
      { init_once =
        { module = "Process"
        , params.cmd =
            "/opt/fabinfra/adapters/initiator_process_example/init.once.py"
        }
      , init_loop =
        { module = "Process"
        , params.cmd =
            "/opt/fabinfra/adapters/initiator_process_example/init.loop.py"
        }
      }
```

```
  , init_connections =
    [ { machine = "zam-raum1-ecke1-lamp", initiator = "init_once"}
    , { machine = "zam-raum1-ecke2-arrow", initiator = "init_loop"}
    ]
```

> Siehe auch Aktor: Generisches Python-Template für "Process"

Version #16
Erstellt: 2025-03-01 12:37:29 CET von Mario Voigt (Stadtfabrikanten e.V.)
Zuletzt aktualisiert: 2025-03-17 20:23:55 CET von Mario Voigt (Stadtfabrikanten e.V.)