Job Java Job

Mit diesem Job können komplexe Java-Programmierungen am Webdesk durchgeführt werden.

Da hierfür Webdesk-Pakete geladen werden müssen ist es empfehlenswert sich hierfür mit Workflow in Verbindung zu setzen.

Beispiel: Java-Job zum Löschen nicht mehr verwendeter Prozessdefinitionen

Dieses Beispiel zeigt ein kommentiertes Job-Skeleton, das als Ausgangspunkt für eine konkrete Implementierung dienen kann. Wichtige Hinweise:

- Vor dem produktiven Einsatz: immer Backup, Probelauf (dry-run) und Test in einer Staging-Umgebung durchführen.
- Die hier gezeigten Klassen und Methoden sind Platzhalter und müssen an die konkrete Webdesk-API / Job-Basisklasse Ihrer Installation angepasst werden.
- Der Job darf nur Prozessdefinitionen löschen, die keine aktiven Instanzen haben und nirgends mehr referenziert werden (Aktionen, Menüeinträge, Prozessreferenzen).

Kurze Ablaufbeschreibung

1. Ermittle alle Prozessdefinitionen, die seit X Tagen/Jahren nicht mehr veröffentlicht/benutzt wurden.
2. Prüfe für jede Definition, ob aktive Instanzen existieren.
3. Prüfe, ob Referenzen (Aktionen, Menü, Prozessreferenzen) auf die Definition verweisen.
4. Wenn dryRun=true: Ergebnisreport erstellen, nichts löschen. Wenn dryRun=false und alle Prüfungen positiv: Definition löschen (oder verschieben/archivieren, falls gewünscht).
5. Report generieren und per Mail verschicken (optional).

Beispiel-Java-Code (Skeleton)

Dieses Beispiel ist als Template gedacht. Bitte passen Sie Paketnamen, Basisklassen und Service-Methoden an Ihre Umgebung an.

```
package at.workflow.webdesk.jobs;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

// TODO: Anpassen an die konkrete Job-Basisklasse / Schnittstelle
// import at.workflow.webdesk.jobs.AbstractWebdeskJob;
// import at.workflow.webdesk.services.ProcessDefinitionService;
// import at.workflow.webdesk.services.ReferenceCheckService;
// import at.workflow.webdesk.services.MailService;

public class CleanupProcessDefinitionsJob /* extends AbstractWebdeskJob */ {
    private static final Logger log =
 LoggerFactory.getLogger(CleanupProcessDefinitionsJob.class);

    // Konfigurationsparameter (können z.B. über Job-Parameter gesetzt werden)
    private boolean dryRun = true; // standardmäßig Probelauf
    private int maxDaysUnused = 365; // Beispiel: seit einem Jahr unbenutzt

    // Platzhalter für Services (in der realen Implementierung über Spring/Bean-Lookup
 beziehen)
    // private ProcessDefinitionService processDefinitionService;
    // private ReferenceCheckService referenceCheckService;
    // private MailService mailService;

    public void execute(/*JobExecutionContext ctx*/) throws Exception {
        log.info("Start CleanupProcessDefinitionsJob (dryRun={})", dryRun);
```

```
        // 1) Prozessdefinitionen ermitteln
        // List<ProcessDefinition> candidates =
 processDefinitionService.findUnusedSince(maxDaysUnused);

        // 2) Iteration und Prüfungen
        // for (ProcessDefinition pd : candidates) {
        //     if (processDefinitionService.hasActiveInstances(pd)) {
        //         log.info("Skipping {} - active instances present", pd.getName());
        //         continue;
        //     }
        //     if (referenceCheckService.isReferenced(pd)) {
        //         log.info("Skipping {} - still referenced by actions/menus",
 pd.getName());
        //         continue;
        //     }
        //     if (dryRun) {
        //         log.info("[DRY-RUN] Would delete process definition: {} (id={})",
 pd.getName(), pd.getId());
        //     } else {
        //         log.info("Deleting process definition: {} (id={})", pd.getName(),
 pd.getId());
        //         processDefinitionService.delete(pd);
        //     }
        // }

        // 3) Report und optional Mailversand
        // String report = buildReport(...);
        // mailService.sendReport("CleanupProcessDefinitionsJob result", report);

        log.info("End CleanupProcessDefinitionsJob");
    }

    // Helper: Beispiel-Setter für Tests
    public void setDryRun(boolean dryRun) { this.dryRun = dryRun; }
    public void setMaxDaysUnused(int days) { this.maxDaysUnused = days; }
}
```

Deployment / Hinweise

- Der kompiliert Jar muss in die Webdesk-Classpath / entsprechende Module integriert werden. Abstimmung mit Workflow EDV ist empfohlen.
- Wenn Sie Spring-Beans verwenden, registrieren Sie die Beans und stellen Sicher, dass der Job zur Laufzeit die notwendigen Services bezieht (z. B. über ApplicationContext oder Konstruktor-Injection).
- Testen Sie ausgiebig mit dryRun=true, bevor Sie die schreibenden Aktionen aktivieren.

Relevante Dokumente

- Job Java Job: https://extranet.workflow.at/daisy/webdesk-manual/7474-dsy.html
- Job Custom Java Script (Scripting-Alternativen / Beispiele): https://extranet.workflow.at/daisy/webdesk-manual/7473-dsy.html
- Job WfHouseKeeping (bereits vorhandener Housekeeping-Job): https://extranet.workflow.at/daisy/webdesk-manual/11315-dsy.html
- Allgemeine Workflow-Jobs / Job-Übersicht: https://extranet.workflow.at/daisy/webdesk-manual/3776-dsy.html

Felder

Name	Wert
Modul	Portal & Organisation (po)
Webdesk Actionname	Java Job

Artefakt-Typ	Job

1. https://extranet.workflow.at/daisy/webdesk-manual/7474-dsy.html
2. https://extranet.workflow.at/daisy/webdesk-manual/7473-dsy.html
3. https://extranet.workflow.at/daisy/webdesk-manual/11315-dsy.html
4. https://extranet.workflow.at/daisy/webdesk-manual/3776-dsy.html