

# Job Custom Java Script

---

## Zweck

Der Job **customJavaScript** führt ein frei definierbares JavaScript im Serverkontext aus. Damit lassen sich einmalige Datenkorrekturen oder wiederkehrende Wartungsaufgaben automatisieren, ohne neue Deployments durchführen zu müssen.

## Ablauf

1. Beim Start liest der Job die hinterlegte Konfiguration ein und prüft, ob der Lauf innerhalb einer Datenbanktransaktion erfolgen soll.
2. Der JavaScript-Quelltext wird aus der Konfiguration entnommen und unverändert an den Webdesk-Scriptingdienst übergeben.
3. Dem Skript stehen automatisch die Objekte logger und log zur Verfügung, sodass Meldungen in die Server-Logs geschrieben werden können.
4. Der Rückgabewert des Skripts wird ignoriert; maßgeblich sind Seiteneffekte und Log-Ausgaben.
5. Der Code wird vollständig ausgeführt; auftretende Fehler werden geloggt und führen zum Abbruch des Jobs.

## Konfiguration

### Javascript Code

Hier wird der komplette Skriptinhalt gepflegt. Der Editor unterstützt Syntax-Highlighting und ermöglicht mehrzeilige Eingaben. Achten Sie auf gültige JavaScript-Syntax und auf idempotente Abläufe, damit wiederholte Ausführungen keine unerwünschten Seiteneffekte verursachen.

### Use Transaction

Schaltet den Joblauf in eine Datenbanktransaktion. Falls aktiviert, werden sämtliche Änderungen erst am Ende übernommen; bei Fehlern erfolgt automatisch ein Rollback. Deaktivieren Sie die Option nur, wenn das Skript bewusst ohne Transaktion laufen soll (z. B. bei externen Aufrufen).

## Erweiterte Nutzung

- **Zugriff auf Java-Klassen:** Über das globale Packages-Objekt können Sie jede Klasse aus dem Webdesk-Paket aufrufen, z. B. var Group = Packages.at.workflow.webdesk.po.model.PoGroup; Anschließend stehen alle öffentlichen Methoden der Klasse im Skript bereit.
- **Spring-Beans beziehen:** Der Scripting-Service stellt automatisch die Variable appCtx bereit (vom Typ ApplicationContextGetBeanAccessor). Über appCtx.getBean("BeanName") erhalten Sie jede registrierte Spring-Bean, z. B. var personService = appCtx.getBean("PoPersonService");.
- **Weitere Hilfsobjekte:** Neben logger, log und appCtx können Sie beliebige eigene Variablen in das Skript einführen, indem Sie sie vor der Ausführung in der Konfiguration oder im Job selbst befüllen.

## Bedienhinweise

- Bewahren Sie Sicherungskopien des Skripts außerhalb der Konfiguration auf, um Änderungen nachverfolgen zu können.
- Nutzen Sie umfangreiche Logausgaben über logger.info(...) und logger.error(...), um das Verhalten nachträglich nachvollziehen zu können.

- Testen Sie komplexe Skripte zunächst in einer Staging-Umgebung und führen Sie dort mehrere Läufe durch, bevor Sie den Job produktiv aktivieren.
- Verwenden Sie Transaktionen für alle Operationen, die mehrere Objekte verändern; so verhindern Sie halb erledigte Änderungen bei Fehlern.
- Planen Sie regelmäßige Reviews, um alte Skripte zu entfernen oder anzupassen, falls sich Datenstrukturen geändert haben.

## Felder

Name	Wert
Modul	Portal & Organisation (po)
Webdesk Actionname	customJavaScript
Artefakt-Typ	Job