



Bundesverwaltungsamt



## IsyFact-Standard

# Detailkonzept Komponente Batch

**Version 3.10**  
**21.03.2016**



„Detailkonzept Komponente Batch“ des Bundesverwaltungsamts ist lizenziert unter einer Creative Commons Namensnennung 4.0 International Lizenz.



„Detailkonzept Komponente Batch“  
des Bundesverwaltungsamts ist lizenziert unter einer  
Creative Commons Namensnennung 4.0 International Lizenz.

Die Lizenzbestimmungen können unter folgender URL heruntergeladen  
werden: <http://creativecommons.org/licenses/by/4.0>

**Ansprechpartner:**

Referat Z II 2  
Bundesverwaltungsamt  
E-Mail: [isyfact@bva.bund.de](mailto:isyfact@bva.bund.de)  
Internet: [www.isyfact.de](http://www.isyfact.de)

## Dokumentinformationen

Dokumenten-ID:	Detailkonzept_Komponente_Batch.docx
----------------	-------------------------------------

## Java Bibliothek / IT-System

Name	Art	Version
plis-batchrahmen	Bibliothek	1.5.x

# Inhaltsverzeichnis

<b>1. Management Summary .....</b>	<b>6</b>
<b>2. Einleitung .....</b>	<b>7</b>
2.1. Aufbau und Zweck des Dokuments .....	7
2.2. Vorgehen .....	7
2.3. Was ist ein Batch? .....	7
2.3.1 Die Bestandteile eines Batches und die Zielarchitektur .....	7
2.3.2 Checkpoint / Restart Logik .....	8
2.3.3 Der Unterschied zwischen Batch-Netz, Batch-Schritt und Batch-Implementierung .....	9
2.3.4 Die Aufteilung in Batchrahmen und Batchlogik .....	10
2.3.5 Grobe Architektur des Batchrahmens .....	10
2.4. Batches als eigenes IT-System .....	11
<b>3. Anforderungen .....</b>	<b>13</b>
3.1. Anforderungen an den Batchrahmen .....	13
3.2. Anforderungen an die Batchimplementierung .....	13
3.3. Anforderungen an die Dokumentation von Batches .....	14
<b>4. Ausgrenzungen .....</b>	<b>15</b>
<b>5. Der Batchrahmen .....</b>	<b>16</b>
5.1. Die Status und Startarten des Batchrahmens .....	17
5.2. Die Konfiguration des Batchrahmens .....	18
5.2.1 Konfigurationsdatei und Kommandozeilen-Parameter .....	18
5.2.2 Betriebliche Konfiguration der Ausführungsbean .....	20
5.2.3 Die Konfiguration der Spring-Kontexte .....	21
5.3. Die Tabellen des Batchrahmens .....	21
5.3.1 Verwaltung der Tabelle .....	22
5.4. Die Transaktionssteuerung .....	23
5.5. Die Restart Funktionalität .....	23
5.5.1 Die Konfiguration im Restart-Fall .....	24
5.5.2 Beenden eines Laufs mit „kill -15“ und Wiederanlauf .....	24
5.5.3 Wiederanlauf nach Abbruch durch die Überschreitung der maximalen Laufzeit .....	24

5.5.4	Wiederanlauf nach Abbruch durch „kill -9“ .....	24
5.5.5	Das Vorlesen durch die Ausführungsbean.....	24
5.6.	Die Überwachungs-Funktionalität .....	25
5.7.	Authentifizierung und Autorisierung .....	26
5.8.	Das Deployment eines Batches .....	27
5.9.	Rückgabewerte des Batchrahmens .....	28
5.10.	Testmodus .....	31
5.10.1	Testmodus mit Rollback.....	32
5.10.2	Testmodus ohne Schreiboperationen .....	32
<b>6.</b>	<b>Die Ausführungsbeans .....</b>	<b>34</b>
6.1.	Keine Transaktionssteuerung in einer Ausführungsbean.....	34
6.2.	Logging, Protokollierung und Statistik-Aufrufe implementieren.....	34
6.3.	Fachliche Logik in den Komponenten der Fachanwendung implementieren .....	34
6.4.	Plausibilitätsprüfung in der Initialisierung .....	34
6.5.	In Initialisierung Schlüssel lesen, Satz-Verarbeitung über Lookups	34
6.6.	Informationen zum Batch-Benutzer bereitstellen .....	35
6.7.	Fehlerbehandlung in Ausführungsbeans durchführen .....	35
6.8.	Beispiele Satzverarbeitung .....	36
6.8.1	Sonderfall Blocklöschung.....	36
6.8.2	Sonderfall eigenständige Transaktionssteuerung .....	37
<b>7.</b>	<b>Ausnahmeregelungen .....</b>	<b>38</b>
<b>8.</b>	<b>Quellenverzeichnis .....</b>	<b>39</b>
<b>9.</b>	<b>Abbildungsverzeichnis .....</b>	<b>40</b>
<b>10.</b>	<b>Tabellenverzeichnis .....</b>	<b>41</b>

## **1. Management Summary**

Gemäß der IsyFact-Referenzarchitektur sind Batches für eine Anwendung neben dem Online-Zugriff eine weitere Zugriffsmöglichkeit auf ihre Fachlichkeit. Da Batches ohne manuelle Eingriffe auskommen sollen, müssen sie Logik für die Behandlung von und das Wiederanlaufen nach Fehlern enthalten. Diese Logik ist in einem Batchrahmen enthalten, welcher keine anwendungsspezifische Logik enthält.

In diesem Dokument wird der von den IsyFact-Standards definierte Batchrahmen sowie die Anforderungen an die anwendungsspezifische Batchverarbeitung beschrieben.

## 2. Einleitung

### 2.1. Aufbau und Zweck des Dokuments

Dieses Dokument beschreibt die Funktionsweise und Besonderheiten von Batches innerhalb der IsyFact Anwendungslandschaft. Zunächst wird auf den Begriff „Batch“ und dessen Anwendungslogik eingegangen. Die Trennung von Batchrahmen und Batchlogik ermöglicht eine Trennung der allgemeinen Funktionsweise von dem fachspezifischen Teil.

In Kapitel 3 werden die Anforderungen, in Kapitel 4 die Ausgrenzungen definiert. Die Funktionsweise des Batchrahmens und die verschiedenen Startarten sowie die Konfiguration folgt in Kapitel 5. Dort werden auch die Transaktionssteuerung und die Restart-Funktionalität beschrieben, die es ermöglicht, abgebrochene Batches fortzusetzen. Die Überwachung mittels JMX ist ebenfalls enthalten. Kapitel 6 beschreibt schließlich, wie die konkrete Batchlogik über Ausführungsbeans angebunden wird.

### 2.2. Vorgehen

Der Begriff „Batch“ bedeutet im Kern lediglich „nicht interaktive Stapelverarbeitung“. Wie ein Batch technisch umgesetzt werden soll, hängt davon ab, was konkret unter einem Batch verstanden wird. Im Folgenden wird deshalb zuerst beschrieben, was ein Batch im Sinne der IsyFact ist.

Ein Batch teilt sich auf in einen „Batchrahmen“, die Batchlogik der einzelnen Batch-Anwendungen, sowie die von ihnen aufgerufenen Komponenten des Anwendungskerns. Was darunter verstanden wird, wird in Kapitel 2.3.4 beschrieben.

### 2.3. Was ist ein Batch?

Eine Fachanwendung stellt über ihre Komponenten fachliche Operationen bereit. Diese Operationen werden (zum Teil) als Services bereitgestellt. Sie werden aber auch bei der automatisierten Bearbeitung größerer Datenmengen in Batches verwendet. Typischerweise verarbeiten Batches dabei Dateilieferungen oder eine definierte Menge von Datensätzen aus dem Datenbestand. Einige der Operationen von Anwendungskomponenten werden nur für Services, einige nur für Batch-Verarbeitungen, andere für beides verwendet.

#### 2.3.1 Die Bestandteile eines Batches und die Zielarchitektur

Ein Batch verwendet für die fachliche Verarbeitungslogik die Operationen der Komponenten der Fachanwendung. Neben der fachlichen Verarbeitungslogik besteht die Anwendungslogik eines Batches aus folgenden Bestandteilen:

- der technischen Logik, welche spezifisch für genau eine Batch-Implementierung ist: Funktionalität für das Einlesen der für die Verarbeitung benötigten Daten, das Aufrufen der Fachkomponenten etc.
- querschnittlicher Logik, welche für alle Batches gleich ist. Dies ist z. B. die Checkpoint-Restart Logik (siehe Kapitel 2.3.2).

In Abbildung 1 werden diese Bestandteile in der Zielarchitektur hervorgehoben: Die spezifische technische Batchlogik wird in Punkt 1 dargestellt, die fachlichen Operationen der Komponenten finden sich in Punkt 2. Die querschnittliche Logik wird lediglich als Bibliothek eingebunden und ist in der Abbildung nicht dargestellt.

Die Komponente „Batchsteuerung“ der Zielarchitektur bezieht sich auf die Steuerung eines Batch-Netzes (s. u.) und wird über UC4 (siehe auch [ÜberwachungKonfigKonzept]) umgesetzt. Sie wird in diesem Dokument nicht betrachtet.

Die Komponente „Batchlogik“ der Zielarchitektur ist eine logische Komponente, welche die technische Logik der Batchimplementierungen der Fachanwendung beinhaltet. Diese Logik ist technisch gesehen keine Komponente, sondern eine Sammlung unabhängiger Klassen in der Batch-Schicht der Fachanwendung. Im Dokument wird unter Batchlogik diese Klassensammlung und keine Komponente verstanden.

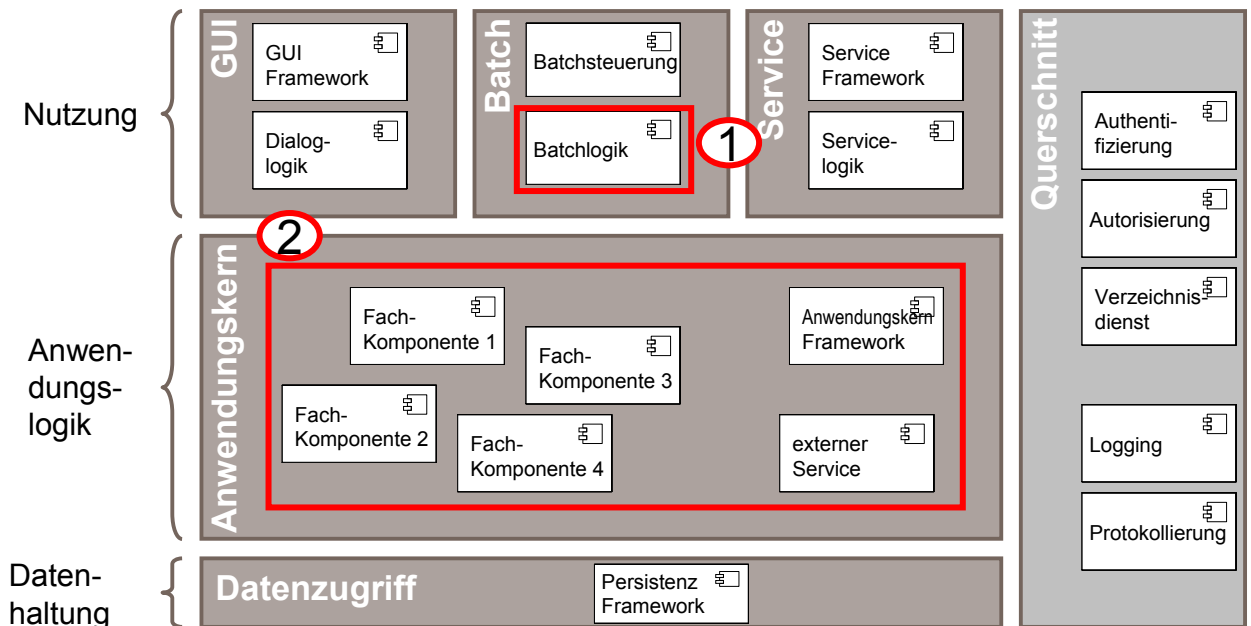


Abbildung 1: Die Anwendungslogik eines Batches (Zielarchitektur)

### 2.3.2 Checkpoint / Restart Logik

Da ein Batch in einem Lauf eine große Anzahl an Datensätzen verarbeiten muss, soll seine Verarbeitung nicht in einer Transaktion durchgeführt werden. Vielmehr wird ein Batch sein Arbeitspaket in mehreren Paketen, d. h. Transaktionen, abarbeiten. Bei einem Wiederanlauf nach einem Fehler muss der Batch in der Lage sein, die bis zu seinem letzten Commit verarbeiteten Datensätze zu überlesen und nur die „neuen“ Datensätze zu verarbeiten. Das Commit einer Transaktion bezeichnet man in diesem Fall als Checkpoint. Das Überlesen alter Datensätze nennt man Checkpoint / Restart Fähigkeit.

Für die Umsetzung der Checkpoint / Restart Fähigkeit muss der Batch in jeder Transaktion speichern, welche Datensätze bis zum Commit verarbeitet wurden. Üblicherweise geschieht das durch die Ablage der Anzahl



verarbeiteter Datensätze bis zum Commit. Abgelegt werden diese Daten in einer Batch-Statustabelle, welche für jeden Batch eine Zeile enthält.

### 2.3.3 Der Unterschied zwischen Batch-Netz, Batch-Schritt und Batch-Implementierung

Die Verarbeitung von Massendaten durch einen Batch verläuft meist in mehreren Schritten: Die Verarbeitung einer Datei kann beispielsweise aus ihrer Übertragung auf einen internen Server, ihrer nachfolgenden Prüfung und der Verarbeitung ihres Inhalts bestehen. Diese Verarbeitungsschritte (*Batch-Schritte*) eines Batches bilden ein „Batch-Netz“. Jeder Schritt wird über den Aufruf eines Shell-Skripts gestartet.

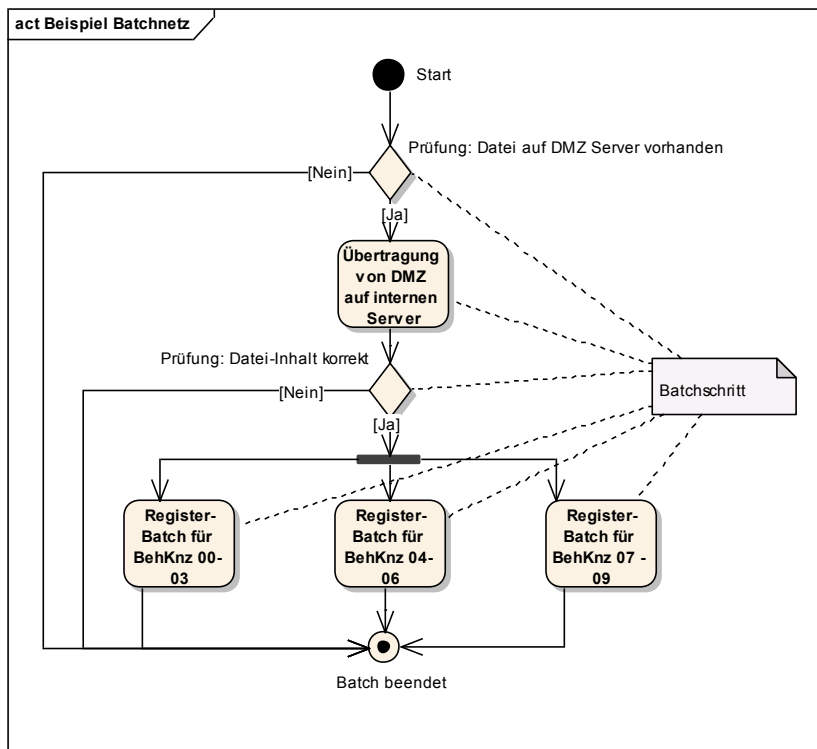


Abbildung 2: Ein Batch-Netz

Die meisten Batch-Schritte sind keine Anwendungen: Sie werden komplett über ein Skript umgesetzt. Nur die Schritte, welche die Verarbeitung in der Fachanwendung durchführen, sind Anwendungen und besitzen eine Java-Implementierung (die *Batchschritt-Implementierung*). In obiger Abbildung wird die Verarbeitung in der Fachanwendung durch drei Batch-Schritte parallel durchgeführt. Alle drei verwenden dieselbe Batchschritt-Implementierung, arbeiten aber auf anderen Daten.

Ein Batch-Netz wird durch einen Job-Scheduler gesteuert. Ein Batch-Schritt ist dabei der Aufruf, welcher vom Scheduler aufgerufen wird. Aufgerufen wird ein Skript mit einer bestimmten Konfiguration. Über dieses Skript wird ggf. eine Fachanwendung mit der übergebenen Konfiguration ausgeführt. Die Konfiguration eines Batchschrittes enthält stets eine Batchschritt-ID, welche den Batchschritt eindeutig identifiziert. Das bedeutet, dass jeder der drei Batchschritte in Abbildung 2 eine eigene Batchschritt-ID erhält, obwohl alle drei Batchschritte die gleiche Batchschritt-Implementierung verwenden. Der

Grund ist, dass jeder Batchschritt auf einem anderen Nummernkreis arbeitet und somit die Batchschritte unterscheidbar sein müssen. Diese Unterscheidung erfolgt über die Batchschritt-ID.

Dieses Dokument befasst sich nicht mit Batch-Netzen oder ihrer Steuerung. Es behandelt die Fachanwendung-Batchimplementierungen und ihre Konfigurationen. Der Einfachheit halber wird deshalb ein Fachanwendung-Batchschritt als „Batch“ und eine Batchschritt-ID als „Batch-ID“ bezeichnet.

Wichtig ist jedoch die Unterscheidung zwischen Batch und Batch-Implementierung: Der Batch ist der Aufruf aus der Produktionssteuerung heraus. Über ihn wird die aufzurufende Batch-Implementierung sowie seine Konfiguration (inklusive der Batch-ID) definiert. Die Implementierung ist Java-Code, der von mehreren Batches verwendet werden darf.

#### **2.3.4 Die Aufteilung in Batchrahmen und Batchlogik**

Das Einlesen und Verarbeiten von Datensätzen durch den Aufruf fachlicher Komponenten wird in diesem Dokument *Batchlogik* genannt. Diese ist spezifisch für genau eine Batch-Implementierung.

Neben dieser Logik enthält eine Batch-Implementierung Querschnittsfunktionalität, welche stets gleich ist. Für eine Batch-Implementierung umfasst diese Querschnittsfunktionalität folgende Punkte:

- Einlesen der Konfiguration des Batches über die Kommandozeile und Konfigurationsdateien
- Instanzieren der Fachanwendungs-Komponenten in einem Spring-Kontext
- Initialisieren, Auslesen und Verwalten der Batch-Statustabelle
- Aufruf des Überlesens von Datensätzen bei einem Restart
- Steuerung der Transaktionen inklusive Speicherung des Fortschritts in einer Status-Tabelle
- Bereitstellung von Überwachungsmöglichkeiten
- Authentifizierung und Autorisierung eines betrieblich konfigurierten Batchbenutzers über die T-Komponente Sicherheit
- Konfiguration, Instanziierung und Aufruf der eigentlichen Batchlogik

Diese Funktionalität wird in einer Batch-Implementierung durch eine querschnittliche Komponente namens *Batchrahmen* umgesetzt. Diese wird über eine Bibliothek bereitgestellt, welche in jede Fachanwendung eingebunden wird.

#### **2.3.5 Grobe Architektur des Batchrahmens**

Für den Batchrahmen wurde folgende grobe Architektur gewählt:

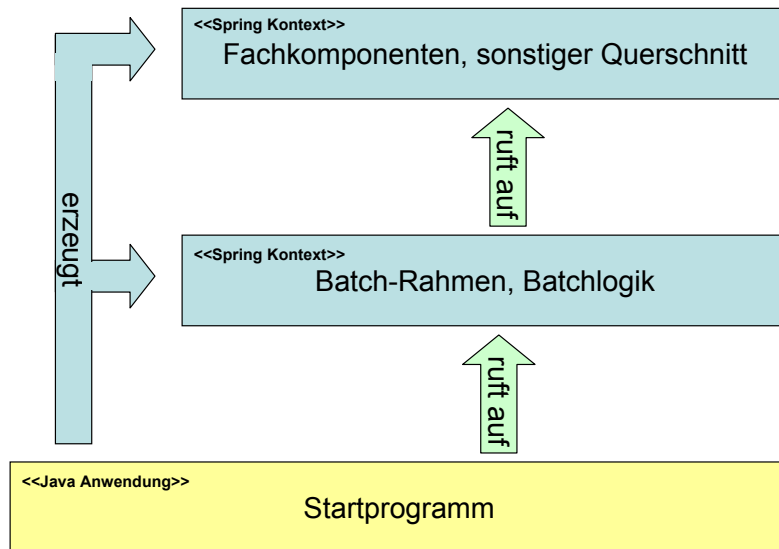


Abbildung 3: Grobe Architektur des Batchrahmens

Der Batchrahmen besteht aus einem Startprogramm, welches notwendige Initialisierungen vornimmt, und einer Komponente Batchrahmen. Die Komponente Batchrahmen übernimmt die Steuerung des Batches und den Aufruf der Batchlogik.

Die Komponenten einer Fachanwendung werden für einen Batch, genau wie in der Webanwendung, über das Spring-Framework verwaltet. Damit die Konfigurationen für die Webanwendung und für die Batch-Implementierungen möglichst gleich sind, werden die Komponenten für den Batchrahmen und die Batchlogik in einer separaten Konfigurationsdatei und einem separaten Spring-Kontext abgelegt. Dieser Kontext ist ein Kind-Kontext des eigentlichen Anwendungskontextes und kann alle Beans des Anwendungskontextes verwenden. So kann die Konfiguration der Webanwendung mit minimalen Anpassungen auch für den Batch verwendet werden.

Der Batchrahmen benötigt für die Speicherung des Fortschritts und des Status der Batches eine Datenbankverbindung. In der Datei werden die vom Batchrahmen benötigten Informationen gespeichert, siehe Kapitel 5.3.

Bei Bedarf können in einem Batch auch weitere Datenbankverbindungen genutzt werden. Die Einbindung weiterer Datenbanken ist in [DatenzugriffDetailkonzept] beschrieben.

## 2.4. Batches als eigenes IT-System

Batches existieren meistens als weiterer Zugangsweg in der Nutzungsschicht einer bestehenden Fachanwendung (siehe Abbildung 1), da sie auf den Daten des jeweiligen Anwendungssystems operieren.

Es kann allerdings auch Batches geben, die unabhängig von einer bestimmten Fachanwendung sind. Dies kann beispielsweise der Fall sein, wenn ein Import-Batch angelieferte Dateien in verschiedene Fachanwendungen importiert. Wenn Batches nicht eindeutig einem

Anwendungssystem zugeordnet werden können, kann man sie auch als eigenständiges IT-System implementieren. Dieses „Batch-System“ kann dann die Service-Schnittstellen anderer Fachanwendungen aufrufen. Insgesamt ist dies jedoch als Sonderfall zu betrachten. In der Regel gehören Batches zu Fachanwendungen, sie enthalten damit den fachlichen Code dieser Anwendung und nutzen diesen nicht über Service-Schnittstellen (siehe Kapitel 5.8).

In diesem Fall ist das Anwendungssystem architektonisch ein eigenständiges IT-System nach Zielarchitektur, nur ohne Service und GUI Bereiche in der Nutzungsschicht.

Batches die zu einer Fachanwendung gehören sollen jedoch nicht als eigenständiges IT-System umgesetzt werden.

### 3. Anforderungen

#### 3.1. Anforderungen an den Batchrahmen

Für den Batchrahmen selbst gelten folgende Anforderungen:

- Er soll den vorhandenen Nutzungsvorgaben und Querschnittskonzepten entsprechen.
- Die Spring-Konfiguration für den Batch soll der Konfiguration der Webanwendung möglichst ähnlich sein.
- Er soll möglichst wenige Anforderungen an (bzw. Annahmen in Bezug auf) die Batchlogik stellen.
- Die Batchlogik soll möglichst einfach implementiert werden können.
- Die Batchrahmen-Implementierung soll programmtechnisch effizient sein.

#### 3.2. Anforderungen an die Batchimplementierung

Für die umzusetzende Batchlogik gelten folgende Anforderungen:

- Sie sollen keine Fachlogik enthalten, sondern die Komponenten des Anwendungskerns verwenden. Abweichungen sind nur aus zwingenden Performance-Gründen erlaubt. Dies ist jeweils mit dem technischen Chefarchitekten abzustimmen und im Systementwurf zu dokumentieren. Falls in Ausnahmefällen direkte Datenbankzugriffe notwendig sind, sind alle weiteren Verarbeitungen an die Fach-Komponenten zu delegieren.
- Die Batchlogik soll die Logging-, Statistik- und Protokollierungsaufgaben nach Anforderung des Projekts umsetzen. Diese wird nicht durch den Batchrahmen umgesetzt. Insbesondere die Statistikinformationen sind für Betrieb und Fachbereich wichtige Informationsquellen, um beispielsweise Laufzeitabschätzungen für die Produktion durchführen zu können. In der Statistik sollte immer enthalten sein, wie viele Sätze verarbeitet wurden (erfolgreich und mit Fehlern) und was konkret unter dem Begriff „Satz“ zu verstehen ist. Ist dies in der Statistik nicht vermerkt, wird davon ausgegangen, dass es sich um den Hauptsatz der Fachanwendung handelt. Was der Hauptsatz einer Fachanwendung ist, hängt von der konkreten Anwendung ab.
- Die Batchlogik soll die Funktionalität des „Überlesens“ von Datensätzen übernehmen. Ob und welche Datensätze zu überlesen sind, teilt der Batchrahmen mit.
- Die Batchlogik soll dem Batchrahmen hinreichende Informationen übergeben, um den Wiederanlaufpunkt bei einem späteren Restart zu bestimmen.
- Der Batchrahmen führt in konfigurierbaren Intervallen Commits durch, um das Fortsetzen des Batches nach einem Abbruch zu ermöglichen. Die Batchlogik soll mit solchen periodischen Commits umgehen können.
- Die Batchlogik sollte – sofern es keine inhaltlichen Abhängigkeiten gibt – mit parallel laufenden Batches umgehen können. Damit ist nicht nur gemeint, dass die gleiche Batchschritt-Implementierung parallel auf z. B. verschiedenen Nummernkreisen arbeiten kann. Vielmehr ist damit gemeint, dass verschiedene Batchschritt-Implementierung parallel ausgeführt werden können. Dazu ist sicherzustellen, dass gemeinsam genutzte Ressourcen nicht unnötig gehalten werden.

- Bei der Ausgabe von vielen Dateien während eines Batchlaufs sind diese strukturiert in Unterverzeichnissen abzulegen. Es ist eine Datei-Namenskonvention für die Batch-Implementierung festzulegen. Die Unterverzeichnisstruktur und die Anzahl der darin enthaltenen Dateien sollen parametrisierbar sein.

### **3.3. Anforderungen an die Dokumentation von Batches**

Für die Dokumentation von Batches gelten neben der lückenlosen und kurzen Beschreibung der durchgeführten Funktionen noch folgende Anforderungen:

- Es muss auf lang laufende Batches hingewiesen werden. Dies ist abhängig von der umgesetzten Batchlogik und der erwarteten Anzahl an zu verarbeitenden Datensätzen. Der Betreiber des Batches soll damit schon im Vorfeld längere Batchlaufzeiten einkalkulieren können.
- Das zu erwartende Laufzeitverhalten bei steigendem Mengengerüst ist anzugeben. Dadurch soll eine Abschätzung geliefert werden, wie sich der Batch skalieren lässt. Analog zum zuvor genannten Punkt soll dies eine Hilfe für den Betreiber zur Kalkulation von Batchlaufzeiten sein.
- Abhängigkeiten von Batches werden dargestellt.

## 4. Ausgrenzungen

Der bereitzustellende Batchrahmen soll ein unkompliziertes und einfach zu verwendendes Framework sein. Seine Funktionalitäten sollen lediglich die Verarbeitung, nicht andere betriebliche Aspekte abdecken. Explizit ausgegrenzt werden deshalb folgende Themenbereiche:

**Ein explizites Handling von Eingabedateien:** Ein möglicherweise erforderliches Dateihandling übernimmt die spezifische Batchlogik. Der Batchrahmen soll kein Wissen darüber besitzen.

**Die Terminierung der Verarbeitung:** Der Batch stellt keine direkten Schnittstellen bereit, um ihn während der Verarbeitung zu beenden. Allerdings muss es möglich sein, einen aktiven Lauf mit dem Signal „kill -15“ definiert zu beenden. Wie die zugehörige Prozess-ID ermittelt wird, ist in den Betriebshandbüchern für die Prozesse zu definieren. Es ist auch möglich, mit dem „laufzeit“-Parameter eine maximale Laufzeit anzugeben, um den Batch nach Überschreitung der angegebenen Laufzeit sich selbst definiert beenden zu lassen.

**Das Scheduling der Verarbeitung:** Das Scheduling wird nicht durch den Batchrahmen, sondern durch die betriebliche Produktionssteuerung durchgeführt.

**Das Prüfen von Vorbedingungen:** Falls für die Ausführung Vorbedingungen gegeben sein müssen (etwa Dateien in Verzeichnissen vorliegen müssen), so liegt deren Prüfung in der Verantwortung des aufrufenden Skripts.

**Das Warten auf Events:** Der Batchrahmen arbeitet eine Reihe von Datensätzen ab und beendet sich danach. Er ist kein Serverprozess, welcher auf bestimmte Events (Dateien in Verzeichnissen, Sätze in Datenbank) wartet, diese verarbeitet und daraufhin weiter wartet.

**Keine parallele Verarbeitung innerhalb eines Batches:** Es ist erlaubt, dass mehrere Java-Prozesse mit der gleichen Batch-Implementierung (jedoch verschiedenen BatchIDs) parallel laufen<sup>1</sup>. Innerhalb einer Verarbeitung wird jedoch stets mit einem Thread gearbeitet. Multithreading innerhalb eines Batches wird nicht unterstützt.

---

<sup>1</sup> Siehe Unterschied zwischen *Batch-Implementierung* und *Batch*, Kapitel 2.3. Parallel laufende Batches sind vor allem sinnvoll, wenn sie über einem aufgeteilten Datenbestand arbeiten und ein Batch pro Teil verwendet wird.

## 5. Der Batchrahmen

Der Batchrahmen ist das Framework, in welches sich die Logik eines konkreten Batches einfügt. Der Batchrahmen ruft (anhand einer Konfiguration) diese Logik auf. Da eine Batch-Anwendung über Spring verwaltet wird, wird die Batchlogik als Spring-Bean konfiguriert, als sogenannte *Ausführungsbean*. Diese Bean wiederum ruft die Fachanwendungs-Komponenten auf, welche die fachliche Logik enthalten.

Für das Vorlage-Register der Vorlage-Anwendung [VorlageAnwendung] zeigt Abbildung 4 eine Auswahl der vorhandenen Beans. Wichtig ist ihre Verteilung auf zwei Anwendungskontexte: Der Batchrahmen und die Ausführungsbeans werden in einem eigenen Kontext konfiguriert. So müssen wenige Anpassungen vorgenommen werden, um aus der Register-Kontext-Konfiguration als Webanwendung die Konfiguration für den Register-Kontext im Batchbetrieb zu erhalten.

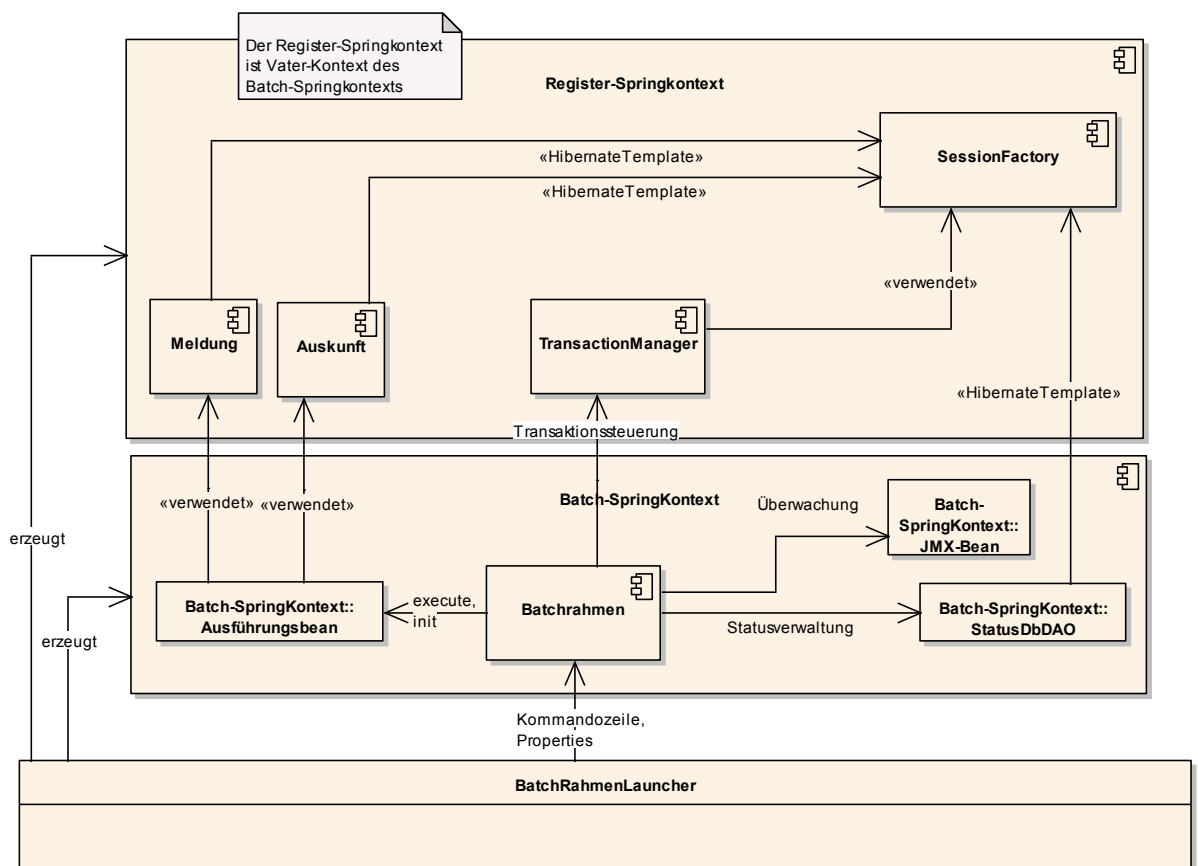


Abbildung 4: Bean-Zugriffe des Batchrahmens

Für neue Batch-Implementierungen einer Fachanwendung müssen ggf. Fachanwendungs-Komponenten angepasst, Ausführungsbeans erstellt und diese im Batch-Springkontext konfiguriert werden. Die folgenden Abschnitte liefern die zur Entwicklung dieser Beans und zur Verwendung des Batchrahmens notwendigen Informationen. Im Einzelnen werden die folgenden Aspekte beschrieben:



- Die Konfigurationsmöglichkeiten des Batchrahmens und der Ausführungsbeans.
- Das Transaktions-Handling, die Restart-Funktionalität und die Status-Tabellen des Batchrahmens.
- Die im Batchrahmen enthaltenen Überwachungsmöglichkeiten.
- Die Authentifizierung und Autorisierung eines Batch-Benutzers.
- Das Deployment der Batches einer Fachanwendung.

### 5.1. Die Status und Startarten des Batchrahmens

Im Folgenden werden die Status beschrieben, in welchen sich ein Batch laut den Batchrahmen-Tabellen befinden kann. Zusätzlich werden, basierend auf diesen Status, die Möglichkeiten zum Starten eines Batchlaufs beschrieben. Diese Informationen werden hier vorgestellt und in den folgenden Abschnitten im Zusammenhang mit den Parametern und Tabellen des Batchrahmens verwendet.

Ein Batch befindet sich in einem von vier Status, welche in den Tabellen des Batchrahmens gespeichert werden:

- **neu:** Status eines noch nicht gelaufenen Batches. Existiert kein Eintrag in der Datenbank, dann wird dieser Status implizit verwendet. Bricht ein noch nicht gelaufener Batch noch in der Initialisierungsphase ab, dann wird dieser Status explizit in die Datenbank geschrieben.
- **laeuft:** Der Batch wurde gestartet und läuft aktuell.
- **abgebrochen:** Der Batch ist
  - mit einem Fehler
  - durch das Signal „kill -15“
  - durch die Überschreitung der konfigurierten maximalen Laufzeit
  - oder weil die Grenze der zu verarbeitenden Datensätze erreicht wurde abgebrochen
- **beendet:** Der Batch ist erfolgreich beendet worden.

Je nachdem, in welchem Zustand sich der Batch laut den Tabellen gerade befindet, lässt er sich nur mit bestimmten Startarten starten. Welche Startart verwendet wird, muss über Parameter definiert werden. Folgende Startarten sind möglich:

- **Batch im Status „neu“ oder „beendet“:**

„start“: Der Batch startet und bearbeitet die Eingabedaten ab dem ersten Datensatz.

- **Batch im Status „laeuft“:**

„ignoriereLauf“: Der Batch wird gestartet, als wäre er erfolgreich beendet worden.

„restart“: Der Batch startet neu und überliest alle bereits verarbeiteten Sätze.

- **Batch im Status „abgebrochen“:**

„restart“: Der Batch startet neu und überliest alle bereits verarbeiteten Sätze.

„ignoriereRestart“: Der Batch startet neu und beginnt die Verarbeitung mit dem ersten Datensatz.

## 5.2. Die Konfiguration des Batchrahmens

### 5.2.1 Konfigurationsdatei und Kommandozeilen-Parameter

Die Konfiguration des Batchrahmens wird über zwei Arten durchgeführt: Über Kommandozeilen-Parameter und über eine Konfigurationsdatei.

Vom Batchrahmen werden folgende Kommandozeilen-Parameter interpretiert:

Parameter	Beschreibung
-cfg <Dateiname>	Name einer Property-Datei mit Konfigurations-Einträgen. Der Dateiname wird relativ zum Klassenpfad interpretiert.
-start	Starten des Batches und Verarbeitung der Daten ab dem ersten Datensatz. Batches im Status „beendet“ <i>müssen</i> über „-start“ gestartet werden. Die gleichzeitige Eingabe von „-start“ und „-restart“ führt zum Abbruch des Batches.
-restart	Starten des Batches nach einem Fehler-Abbruch: Überlesen der bereits verarbeiteten Datensätze. Batches im Status „Abbruch“ müssen mit „-restart“ gestartet werden oder zusätzlich den Parameter - ignoriereRestart enthalten.
-ignoriereRestart	Auch bei Fehlern Start akzeptieren, nicht auf Restart beharren.
-ignoriereLauf	Auch bei Status "laeuft" Start akzeptieren.

Parameter	Beschreibung
-testmodus	Startet den Batch im Testmodus. Dieser arbeitet analog zum normalen Wirkbetrieb, jedoch werden keine Änderungen an Datenbeständen vom Anwendungssystem oder Nachbarsystemen durchgeführt. Ein detailliertes Konzept ist in Kapitel 5.10 beschrieben.
-laufzeit <Minuten>	Gibt eine maximale Laufzeit in Minuten an. Wird die angegebene Zeit überschritten, wird der aktuelle Datensatz zu Ende bearbeitet. Der Batch bricht vor der Verarbeitung des nächsten Datensatzes mit einem dedizierten Return-Code ab.

*Tabelle 1: Kommandozeilen-Parameter des Batchrahmens*

Es können auch andere Parameter angegeben werden, diese müssen die Form -<Parametername> <Parameterwert> besitzen. Diese werden der Konfiguration der Property-Datei hinzugefügt und überschreiben vorhandene Einträge. Solche Parameter müssen nach den Standard-Kommandozeilen-Parametern angegeben werden.

Aus der Property-Datei werden durch den Batchrahmen folgende Properties gelesen:

Property	Beschreibung
Batchrahmen.BeanName	Name der Batchrahmen-Bean
Register.SpringDateien.<N>	Namen der Spring-Konfigurationsdateien der Fachanwendung
Batchrahmen.SpringDateien.<N>	Namen der Spring-Konfigurationsdateien des Batchrahmens
Batchrahmen.Log4JConfigFile	Pfad zur Log4J Konfigurationsdatei
Batchrahmen.CommitIntervall	Anzahl Satz-Verarbeitungen pro Commit
AusfuehrungsBean	Name der Ausführungsbean für die Batchlogik
BatchId	ID des Batches (ID des Batch-Status-Datensatzes)
BatchName	Name des Batches in der Batch-Statustabelle

Property	Beschreibung
Batchrahmen.AnzahlZuVerarbeitendeDatensätze	Falls nicht die ganze Datei verarbeitet werden soll, sondern nur eine gewisse Anzahl an Datensätzen.
Batchrahmen.Ergebnisdatei	Pfad zur XML-Ergebnisdatei des Batchrahmens (siehe Kapitel 5.9)

*Tabelle 2: Die Properties des Batchrahmens*

Die Property-Datei darf beliebige weitere Properties enthalten. Alle Properties (und damit Kommandozeilen-Parameter) werden an die Ausführungsbean übergeben und können genutzt werden, um sie zu konfigurieren.

Die Property-Datei wird als statische Konfiguration abgelegt und kann daher nicht vom Betrieb angepasst werden. Betriebliche Konfigurationen müssen wie in Kapitel 5.2.2 beschriebenen umgesetzt werden.

Dateien, egal ob für Kommandozeilenparameter oder für Properties, müssen mit absoluten Pfaden angegeben werden.

### 5.2.2 Betriebliche Konfiguration der Ausführungsbean

Sämtliche obigen Parameter müssen vom Betrieb nicht angepasst werden. Falls im Ausnahmefall die Batch-ID angepasst werden muss, kann dies über den Kommandozeilen-Parameter `-BatchId <BatchId>` geschehen. Deshalb ist die Konfigurationsdatei nicht im für den Betrieb vorgesehenen config-Verzeichnis, sondern im resources-Verzeichnis abgelegt.

Falls für die Ausführungsbean eines Batches Konfigurationen notwendig sind, welche durch den Betrieb gepflegt werden müssen, so ist dies auf eine von zwei Arten umzusetzen:

- Die Konfigurationen können der betrieblichen Fachanwendung-Konfiguration<sup>2</sup> hinzugefügt werden. Die Ausführungsbean kann dann die Fachanwendung-Konfigurationsbean per Dependency Injection erhalten und sich darüber konfigurieren.

Diese Möglichkeit ist zu verwenden, falls nur ein Batch für die Fachanwendung umgesetzt wird. Falls mehrere Batches umgesetzt werden, ist sie dann zu verwenden, wenn sich die Konfigurationen für die einzelnen Batches nicht widersprechen (also für verschiedene Batches verschiedene Werte für die gleiche Property erwartet werden).

- Die Konfiguration kann in einer neuen Datei abgelegt werden, welche nur für diesen Batch verwendet wird. Diese Datei kann als Properties-Bean geladen und der Ausführer-Bean per Dependency Injection

---

<sup>2</sup> Die im Ordner „config“ liegenden, durch den Betrieb pflegbaren Konfigurationsdateien.

übergeben werden.

Diese Möglichkeit ist zu verwenden, falls für verschiedene Batches verschiedene Konfigurationsdateien benötigt werden.

### 5.2.3 Die Konfiguration der Spring-Kontexte

Wie in Kapitel 2.3.5 beschrieben, werden für einen Batch zwei Spring-Kontexte erzeugt:

- Ein Kontext mit den Beans der eigentlichen Fachanwendung.
- Ein Kontext mit der Batchrahmen-Bean, der Batchrahmen JMX-Bean sowie den Ausführungsbeans für die Batches der Fachanwendung.

Für den Kontext der eigentlichen Fachanwendung können die Spring-Konfigurationsdateien übernommen werden. In ihnen müssen folgende Anpassungen vorgenommen werden:

- Die Beans für Axis-Skeletons und HttpInvoker-Beans sollten entfernt werden.
- Die JMX-Technikbeans müssen entfernt werden. Die eigentlichen Mbeans müssen bleiben.
- Die Konfigurationsdatei für die Batchrahmen-Tabellen (im Vorlage-Register Datei resources/batch/rahmen/hibernate-mapping.xml) ist der sessionFactory-Bean hinzuzufügen.
- Das Resource-Bundle nachrichten-batchrahmen muss der messageSource-Bean hinzugefügt werden.

Die Spring-Konfiguration für den Kontext des Batchrahmens muss neu erstellt werden. Hierfür nehme man Datei resources/batch/rahmen/Batchrahmen.xml des Vorlage-Registers und passe die Konfiguration der Ausführungsbeans an.

### 5.3. Die Tabellen des Batchrahmens

Der Batchrahmen benötigt für seine Checkpoint / Restart Funktionalität die Möglichkeit, bei jedem Commit den aktuellen Stand des Batches in einer Tabelle zu speichern. Dies wird über die Tabelle BatchStatus umgesetzt.


BATCHSTATUS		
	BATCHID	VARCHAR2 (255)
	BATCHNAME	VARCHAR2 (255)
	BATCHSTATUS	VARCHAR2 (255)
	SATZNUMMERLETZTESCOMMIT	NUMBER (19)
	SCHLUESSELLETZTESCOMMIT	VARCHAR2 (255)
	DATUMLETZTERSTART	TIMESTAMP(6) (11)
	DATUMLETZTERABBRUCH	TIMESTAMP(6) (11)
	DATUMLETZTERERFOLG	TIMESTAMP(6) (11)

Abbildung 5: Tabellen-Schema für die Batchrahmen-Tabelle

Die Tabelle BatchStatus enthält für jeden Batch eine Zeile. Ein Batch ist nicht gleichzusetzen mit der Batch-Ausführungsbean. Für eine Bean darf es

mehrere Batches geben, welche die Bean für die Ausführung jeweils anders konfigurieren. Für einen Batch werden folgende Informationen verwendet:

Spalte	Beschreibung
BatchId	Der Schlüssel für den Batch. Die ID sollte aus einem gemeinsamen Präfix für die Fachanwendung, gefolgt von einem Suffix für den konkreten Batch, bestehen.
BatchName	Ein kurzer informativer Name des Batches.
BatchStatus	Einer der folgenden Werte: <b>laeuft</b> : Der Batch wurde gestartet und läuft aktuell. <b>abgebrochen</b> : Der Batch ist abgebrochen und sollte per RESTART neu gestartet werden. <b>beendet</b> : Der Batch ist erfolgreich beendet worden.
Satznummer LetztesCommit	Die Anzahl an Sätzen, welche beim letzten Commit verarbeitet worden sind. Dies wird für die Umsetzung der Restart-Funktionalität verwendet.
Schlüssel LetztesCommit	Der Datenbank-Schlüssel des Datensatzes, der als letztes vor dem Commit bearbeitet wurde. Dies wird für die Umsetzung der Restart-Funktionalität verwendet. Falls der Schlüssel ein zusammengesetzter Schlüssel ist, müssen sämtliche Schlüsselteile (durch Trennzeichen getrennt) in dieses Feld geschrieben werden.
DatumLetzterStart	Das Datum des Starts des aktuellen Batch-Laufs (falls der Status „laeuft“ ist) bzw. des letzten Laufes.
DatumLetzterAbbruch	Das Datum des letzten fehlerhaften Abbruchs des Batches.
DatumLetzterErfolg	Das Datum des letzten erfolgreichen Abschlusses des Batches.

*Tabelle 3: Das Schema der Tabelle BatchStatus*

Die Tabelle liegt im Datenbank-Schema der eigenen Fachanwendung. Es darf keine übergreifenden Tabellen oder ein übergreifendes Datenbankschema für alle Batches gemäß IsyFact-Architektur geben.

### 5.3.1 Verwaltung der Tabelle

Die Tabelle dient nicht der Steuerung des Batches über den Betrieb, sondern nur der Ablage von Informationen zwischen zwei Batch-Läufen. Die Befüllung der Tabelle wird deshalb komplett über den Batchrahmen durchgeführt: Es müssen keine Datensätze manuell befüllt werden: Ist ein Datensatz für einen Batch noch nicht vorhanden, wird er angelegt.

Üblicherweise ist das Locking-Verhalten in Fachanwendungen optimistisch: Datensätze werden nicht explizit gelockt. Stattdessen wird über Versions-Attribute zum Commit-Zeitpunkt geprüft, ob der Datensatz innerhalb der Transaktion verändert wurde. Für die Tabelle des Batchrahmens wird *nicht* optimistisch, sondern pessimistisch gelockt. Zusätzlich wird als Hibernate Locking-Strategie LOCK\_NOWAIT verwendet: Falls auf einen Datensatz zugegriffen wird, welcher gerade gelockt ist, wird nicht bis zur Freigabe gewartet, sondern eine Exception geworfen. Die parallele Ausführung zweier Batches mit gleicher Batch-ID ist nicht erlaubt und soll zum Fehler führen.

#### 5.4. Die Transaktionssteuerung

In der Property-Datei des Batchrahmens wird die Commit-Rate für den Batch über eine Property konfiguriert. Die Transaktionssteuerung für einen Batch arbeitet daraufhin folgendermaßen:

- Die Spring Kontexte werden außerhalb einer Transaktion instanziiert. Danach wird die Kontrolle von der BatchLauncher<sup>3</sup>-Klasse an die Batchrahmen-Bean weitergegeben.
- Die Batchrahmen-Bean startet eine erste Transaktion. Sie aktualisiert die Batchrahmen-Tabellen und initialisiert die Ausführungsbean in dieser Transaktion. Die Ausführungsbean führt im Rahmen dieser Transaktion das „Vorlesen“ bis zum letzten Checkpunkt sowie ggf. nötige Initialisierungen durch. Die Transaktion wird beendet.
- Für die Satz-Verarbeitung wird eine neue Transaktion gestartet.
- Die einzelnen Datensätze werden verarbeitet. Bei Erreichung eines Checkpunkts wird die Status-Tabelle aktualisiert, die Transaktion committed und eine neue gestartet.
- Sobald der letzte Datensatz verarbeitet wurde, wird die Status-Tabelle aktualisiert, auf der Ausführer-Bean eine shutdown-Methode aufgerufen und die (letzte) Transaktion beendet.

Das Verhalten in Fehlerfällen ist zu jedem Zeitpunkt während der Verarbeitung gleich: Der Fehler wird geloggt und die Transaktion zurückgerollt. Danach wird versucht, eine neue Transaktion zu starten, um die Status-Tabelle zu aktualisieren: Der Status wird auf „abgebrochen“ gesetzt und die Spalte DatumLetzterAbbruch auf den aktuellen Zeitpunkt. Daraufhin wird versucht, diese Transaktion zu committen.<sup>4</sup>

#### 5.5. Die Restart Funktionalität

Falls ein Batch durch einen Fehler oder durch das Erreichen der Anzahl zu verarbeitender Datensätze abgebrochen ist, muss ein Restart für ihn durchgeführt werden. In diesem Fall müssen alle bereits verarbeiteten Datensätze übersprungen werden. Ebenso wird die Wiederanlauffähigkeit nach manuellem Terminieren durch das Signal „kill -15“ sichergestellt.

---

<sup>3</sup> Die Start-Anwendung, welche die Kommandozeile interpretiert die Spring-Kontexte erzeugt.

<sup>4</sup> Schlägt die Aktualisierung auf „Abbruch“ der Statustabelle fehl, steht weiterhin „Laeuft“ als Status in der Tabelle. Der nächste Lauf muss dann mit „-ignoreiereLauf“ gestartet werden.

### **5.5.1 Die Konfiguration im Restart-Fall**

Im Restart-Fall wird nicht überprüft, dass die übergebenen Parameter denen entsprechen, die im ursprünglichen Lauf übergeben wurden. Es muss daher bei einem Restart manuell darauf geachtet werden, dass die gleichen Parameter übergeben werden.

### **5.5.2 Beenden eines Laufs mit „kill -15“ und Wiederanlauf**

Durch Senden des Signals „kill -15“ kann ein aktiver Batchlauf beendet werden. Beim Empfang des Signals wird der aktuelle Datensatz zu Ende bearbeitet und der Batch terminiert im wohldefinierten Zustand. In der Statustabelle ist dann der Status „abgebrochen“ vermerkt und der Batchlauf kann mit dem Parameter „-restart“ wieder aufgesetzt werden.

### **5.5.3 Wiederanlauf nach Abbruch durch die Überschreitung der maximalen Laufzeit**

Falls der Batch mit dem „laufzeit“-Parameter gestartet wurde, wird nach Überschreitung der angegebenen maximalen Laufzeit der aktuelle Datensatz zu Ende bearbeitet und der Batch terminiert im wohldefinierten Zustand. In der Statustabelle ist dann der Status „abgebrochen“ vermerkt und der Batchlauf kann mit dem Parameter „-restart“ wieder aufgesetzt werden.

### **5.5.4 Wiederanlauf nach Abbruch durch „kill -9“**

Durch Senden des Signals „kill -9“ wird der aktive Batchprozess von Betriebssystemseite beendet. Dabei wird der Java-Prozess direkt entfernt ohne die Möglichkeit, den Batchrahmen definiert zu terminieren. Dies ist daher nur in Ausnahmesituationen vom Betrieb durchzuführen. Da der Batchrahmen in dieser Situation kein Status-Update mehr schreiben kann, befindet sich der Eintrag „läuft“ in der Statustabelle. Ein Wiederanlauf in dieser Situation ist mit dem Parameter „-restart“ möglich.

### **5.5.5 Das Vorlesen durch die Ausführungsbean**

Um den Batch nach einem Fehler zum nächsten zu bearbeiteten Datensatz „vorlesen“ zu lassen, muss dieser Datensatz identifiziert werden. Dies geschieht zum einen durch den Batchrahmen selbst, welcher die Anzahl der bereits verarbeiteten Datensätze speichert. Bei auf Datenbank-Queries basierenden Batches kann diese Anzahl jedoch ggf. nicht verwendet werden, da sie sich im Laufe der Zeit ändert. Hier ist es notwendig, die zu verarbeitenden Sätze nach ihrem Schlüssel zu sortieren und den als letztes bearbeiteten Schlüssel zu speichern.

Die Batchrahmen Status-Tabelle enthält deshalb zwei Felder: Ein Feld für die Anzahl verarbeiteter Sätze und ein Feld für den Schlüssel des letzten verarbeiteten Datensatzes. Dieser Schlüssel wird von der Ausführungsbean nach der Verarbeitung eines Satzes zurückgegeben.

Bei einem Restart wird der Ausführungsbean in der Initialisierungsmethode übermittelt, ob es sich um einen Restart handelt und welche Werte für den Schlüssel und die Satznummer in der Datenbank stehen. Der Bean ist es



überlassen, das Vorlesen effizient durchzuführen (etwa durch die Aufnahme des Schlüssels in das Selektionskriterium einer Query).

## 5.6. Die Überwachungs-Funktionalität

Die Verarbeitung eines Batches soll überwacht und nachverfolgt werden können. Für die Nachverfolgung können durch die Verarbeitungsbean zu verschiedenen Zeitpunkten Log-, Statistik- oder Protokoll-Einträge erstellt werden:

- Zu Beginn des Batches, während der Initialisierung der Ausführungsbean.
- Nach dem Schreiben jedes Checkpoints.
- Bei der erfolgreichen Beendigung des Batches.

Für die Überwachung wird durch den Batchrahmen eine JMX-Bean bereitgestellt. Über folgende Konfiguration wird ein JMX Agent erzeugt, welcher die Bean nach außen zugreifbar macht:

```
-Dcom.sun.management.jmxremote  
-Dcom.sun.management.jmxremote.port=<PortNummer>  
-Dcom.sun.management.jmxremote.ssl=false  
-Dcom.sun.management.jmxremote.authenticate=true
```

Über die Java Management Console kann man danach auf die Daten zugreifen:

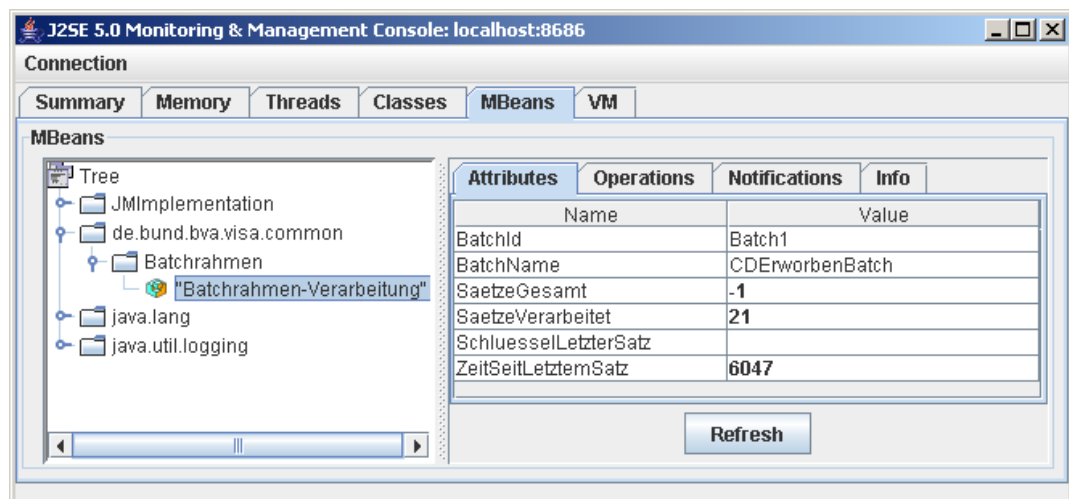


Abbildung 6: Die JMX Überwachung eines Batches

Bereitgestellt werden folgende Informationen:

Property	Beschreibung
SaetzeGesamt	Die Gesamtanzahl der zu bearbeitenden Sätze. Falls diese nicht bekannt ist: -1
SaetzeVerarbeitet	Die Anzahl bereits verarbeiteter Sätze.
SchluesselLetzterSatz	Schlüssel des letzten verarbeiteten Satzes.
ZeitSeitLetztemSatz	Zeitraum in Millisekunden, der bereits für den aktuellen Satz benötigt wurde.
BatchId	Die ID des aktuellen Batches
BatchName	Name des aktuellen Batches.

*Tabelle 4: JMX-Attribute*

Zur Erhöhung der Sicherheit in der Betriebsumgebung muss eine Absicherung der RMI-Schnittstelle für JMX per Benutzername und Passwort erfolgen (siehe JVM-Parameter oben). Hierzu sind die JMX-Benutzer und Passwörter entweder direkt in der Datei

```
JRE_HOME/lib/management/jmxremote.password
```

zu konfigurieren, oder der Ort dieser Datei ist über den JVM-Parameter

```
com.sun.management.jmxremote.password.file
```

anzugeben. Die Datei benötigt einen Eintrag für die Rolle controlRole oder für eine beliebige Rolle, für welche in Datei `JRE_HOME/lib/management/jmxremote.access` readwrite-Zugriff erlaubt ist.

## 5.7. Authentifizierung und Autorisierung

Ein Batch wird innerhalb der Plattform gestartet. Dazu verwendet der Batch die Benutzererkennung eines technischen Benutzers „internes System“, die in seiner Startkonfiguration (als Konfigurationsdatei) hinterlegt und im Benutzerverzeichnis vorhanden ist.

Der Batchrahmen nutzt die T-Komponente Sicherheit (siehe [SicherheitNutzerdok]), um den anfragenden Benutzer über den Access-Manager der Plattform zu authentifizieren. Der Access-Manager authentifiziert den Benutzer anhand der angegebenen Benutzererkennung und des Passwortes, registriert alle Informationen zum Benutzer in der T-Komponente AufrufKontextVerwalter und schließt die Session des Benutzers im Access-Manager.

Im Zuge des Batchlaufs können nun über die T-Komponente Sicherheit Berechtigungsprüfungen (z.B. im Anwendungskern) stattfinden.

Der Batchrahmen fordert vom BatchAusführungsBean die Implementierung der Methode: `getAuthenticationCredentials()`, in der dem Batchrahmen bekanntgegeben wird, mit welchem Benutzer der Batch laufen soll.

```
@Override
public BatchAuthenticationCredentials getAuthenticationCredentials(
    BatchKonfiguration konfiguration);
```

Folgende Informationen zum Benutzer sind anzugeben

Property	Beschreibung
Bhknz	Das Behörden- oder Organisationskennzeichen, der Behörde oder Organisation, der der Benutzer angehört.
Benutzer	Die Kennung des technischen Benutzers (internes System)
Passwort	Das Kennwort des Benutzers

*Tabelle 5: Die Eigenschaften des Batchbenutzers*

In der Regel wird der Benutzer aus der betrieblichen Konfiguration der Anwendung oder alternativ aus den Aufrufparametern beim Batchstart gelesen.

In Ausnahmefällen ist es auch möglich, einen Batch zu implementieren, der ohne Benutzer laufen soll. Dies ist nur möglich, wenn bei Aufrufen des Anwendungskerns keine Autorisierungsprüfungen stattfinden und auch keine Nachbarsystemschnittstellen aufgerufen werden. Soll der Batch ohne Benutzer laufen, so ist in der Methode `null` zurückzugeben.

Enthält eine Anwendung mehrere Batches, so enthält sie auch mehrere BatchAusführungsBeans und kann jeden Batch mit einem eigenen Benutzer ausstatten.

## 5.8. Das Deployment eines Batches

Die Batches werden nicht einzeln deployed. Stattdessen wird ein Paket angeboten, welches sämtliche Batches der Fachanwendung enthält. In der Vorlage-Anwendung geschieht dies durch das Maven-Projekt `cd-register-batch`.

Das erstellte Paket enthält den Code des Batchrahmens sowie den Code der eigentlichen Fachanwendung inklusive der Batch-Ausführungsklassen und der benötigten nicht-betrieblichen Batch-Konfigurationsdateien. Für jeden Batch wird ein Shellskript zum Start bereitgestellt.

Für die Shellskripte existieren keine Vorgaben. Falls vor der Ausführung des Batches Vorbedingungen gelten müssen (etwa Dateien in Verzeichnissen vorliegen sollen), so können sie in diesen Skripten geprüft werden. Beispiele für Aufruf-Skripte von Batches befinden sich in der Bibliothek `plis-batchrahmen`, im Verzeichnis:

```
plis-batchrahmen/src/main/skripte/bin/
```

Eine sinnvolle Aufteilung bei den Shellskripten ist es, ein technisches Startskript zu erstellen, was von den eigentlichen Batch-Shellskripten zum Aufruf genutzt wird. Beispiel für ein solches Startskript befinden sich in der Bibliothek `plis-batchrahmen`, in der Datei:

```
plis-batchrahmen/src/main/skripte/bin/batch-  
ausfuehren.sh
```

Dieses Startskript soll in der Batch-Anwendung unter `<batch-projekt>/src/main/resources/bin/` übernommen werden und mit der richtigen JAVA Version versehen werden.

Ein beispielhaftes Shellskript zum Aufruf eines Batches kann dieses Skript dann nutzen und folgendermaßen aussehen:

```
#!/bin/bash

#
# Parameter für den Erinnerung-Batch
#
# -Testmodus <true|false> Flag ob nur simuliert wird oder
nicht (optionaler Parameter)
#

BATCHDIR=`dirname $0`
$BATCHDIR/batch-ausfuehren.sh -start -cfg
/resources/batch/batch-erinnerung-config.properties -
Batchrahmen.Ergebnisdatei /tmp/erinnerung_out.xml $1 $2
```

#### Listing 1: Beispielhaftes Startskript für einen Batch

Bei der betrieblichen Konfiguration wird nicht zwischen dem Deployment als Web-Anwendung und dem Deployment als Batch unterschieden: Es werden jeweils dieselben betrieblichen Konfigurationsdateien verwendet. Damit der Betrieb Anpassungen dieser Dateien nicht zweimal durchführen muss, wird die betriebliche Konfiguration mit der Web-Anwendung deployed. Vor dem Deployment des Batch-Pakets muss die Webanwendung deployed sein. Falls der Batch nicht auf dem Server der Webanwendung läuft, muss die Konfigurationsdatei der Webanwendung vor der Installation des Batches auch auf diesem Server verfügbar gemacht (z.B. gemountet oder kopiert) werden. Beim Deployment des Batch-Pakets werden über symbolische Links diese betriebliche Konfigurationsdatei referenziert.

### 5.9. Rückgabewerte des Batchrahmens

Der Batchrahmen endet mit einem Returncode und erzeugt optional zusätzlich noch ein ausführliches Verarbeitungsergebnis in Form einer Ergebnisdatei im XML-Format. Die Ergebnisdatei ist eine **fachliche Datei**, sie enthält keine betrieblichen Informationen. Alle betrieblichen Informationen über Ausführung des Batches werden in die Logdatei geschrieben, so dass der Betrieb nur diese Datei betrachten muss. Es gibt keine weiteren betriebsrelevanten Dateien neben der Logdatei.

Der Pfad der Ergebnisdatei wird über einen Konfigurationsparameter des Batches festgelegt (siehe Kapitel 5.2.1). Ist dieser Parameter nicht vorhanden, so wird auch keine Ergebnisdatei geschrieben. Die Ergebnisdatei hat den folgenden Aufbau:

XML-Pfad	Attribut	Typ	Bedeutung
Batch-Ergebnis			Root-Tag des Batch-Ergebnisses
Batch-Ergebnis / Start	Datum	Text	Datum des Starts des Batchlaufs
Batch-Ergebnis / Start	Uhrzeit	Text	Uhrzeit des Starts des Batchlaufs
Batch-Ergebnis / Start	BatchID	Text	Die Batch-ID des Batchlaufs
Batch-Ergebnis / Start	Parameter	Text	Die Parameter des Batchlaufs
Batch-Ergebnis / Meldungen			Liste von Meldungen (Fehler, Warnungen oder Informationen)
Batch-Ergebnis / Meldungen / Meldung	ID	Text	ID des Eintrags (z.B. Fehlernummer)
Batch-Ergebnis / Meldungen / Meldung	Typ	Text	Typ des Eintrags: F, W, I
Batch-Ergebnis / Meldungen / Meldung	Text	Text	Text des Eintrags, z.B. Fehlertext
Batch-Ergebnis / Meldungen / Meldung	Satz	Text	Der fachliche Schlüssel des Hauptsatzes, dessen Verarbeitung diese Meldung verursacht hat. Es ist zu beachten, dass ein technischer Schlüssel an dieser Stelle nicht ausreichend ist, da der Fachbereich und der Betrieb mit diesem technischen Schlüssel nichts anfangen können. Falls es nicht möglich ist den fachlichen Schlüssel des Hauptsatzes an dieser Stelle zu ermitteln bzw. wenn dies nur auf Kosten der Laufzeit möglich ist, dann ist mit dem Fachbereich und dem Betrieb frühzeitig zu klären, ob an dieser Stelle ein anderer

XML-Pfad	Attribut	Typ	Bedeutung
			Schlüssel ausgegeben werden kann.
Batch-Ergebnis / Statistik			Liste von statistischen Daten die während des Batchlaufs ermittelt wurden.
Batch-Ergebnis / Statistik / Statistik-Eintrag	ID	Text	ID des Statistik-Eintrags
Batch-Ergebnis / Statistik / Statistik-Eintrag	Text	Text	Klartext des Eintrags, z. B. „Anzahl gelöschter Datensätze“
Batch-Ergebnis / Statistik / Statistik-Eintrag	Wert	Text	Statistischer Wert, z. B. Anzahl der gelöschten Datensätze
Batch-Ergebnis / Ende	Datum	Text	Datum des Ende des Batchlaufs
Batch-Ergebnis / Ende	Uhrzeit	Text	Uhrzeit des Ende des Batchlaufs
Batch-Ergebnis / Return-Code	RC	Text	Return-Code der Batch-Verarbeitung
Batch-Ergebnis / Return-Code	Text	Text	Den Return-Code zugeordneter Text

Tabelle 6: Format der Ergebnisdatei

Beispiel für ein Batchprotokoll:

```
<?xml version="1.0" encoding="UTF-8"?>
<Batch-Ergebnis>
  <Start BatchID="GesamtauskunftBatch" Datum="2010-07-27" Uhrzeit="14:59:32"
  Parameter="-start - /gesamtauskunft--config.properties -Batchrahmen.Ergebnisdatei
  /Batchprotokoll_Gesamtauskunft.log.xml -Operationsmodus auskunft -DeltaVon 0"/>
  <Meldungen>
    <Meldung ID="" Typ="I" Text="Beginn des Batchlaufes: Gesamtauskunft-Batch."/>
    <Meldung ID="1" FachlicheID="123456789001" Typ="I" Text="Für die CD-Nummer wurde
    eine Auskunft ausgelöst."/>
    <Meldung ID="2" FachlicheID="123456789002" Typ="I" Text="Für die CD-Nummer wurde
    eine Auskunft ausgelöst."/>
  </Meldungen>
  <Statistik>
    <Statistik-Eintrag ID="ANZAHL AUSKÜNFTE" Text="Anzahl der Auskünfte" Wert="2"/>
    <Statistik-Eintrag ID="EROLGREICHE AUSKÜNFTE" Text="Anzahl erfolgreicher Auskünfte"
    Wert="2"/>
  </Statistik>
  <Ende Datum="2010-07-27" Uhrzeit="15:00:01"/>
  <Return-Code RC="0" Text="Verarbeitung ohne Fehler durchgeführt."/>
</Batch-Ergebnis>
```

Zur Auswertung der Ergebnisdatei können XSLT-Stylesheets verwendet werden, die die Ergebnisdatei in eine Textdatei bzw. in HTML umwandeln. Es handelt sich hierbei um eine fachliche Transformation der Daten mit dem Ziel, diese für den Fachbereich zu filtern, zu aggregieren oder in einem bestimmten Format zu Weiterverarbeitung bereitzustellen.

Wenn die Verarbeitung erfolgreich beendet wurde, endet der Batchrahmen mit dem Return-Code 0. Er endet mit einem anderen Return-Code, falls der Batch mit einem Fehler beendet wurde oder gar nicht gestartet werden konnte. Bei Fehlern in den „Ausführungsbeans“ kann der zurückzugebende Return-Code über die aufgetretenen Exceptions bestimmt werden (siehe Kapitel 6.6).

Pro Batch müssen die möglichen Rückgabewerte definiert werden. Die folgenden Werte sind reserviert und müssen von jedem Batch zurückgegeben werden, falls ein entsprechendes Ereignis eintritt:

Return-Code	Bedeutung
0	Verarbeitung ohne Fehler durchgeführt
1	Verarbeitung mit Fehlern durchgeführt
2	Verarbeitung mit Fehlern abgebrochen
3	Batch konnte wegen Fehlern in den Aufrufparametern nicht gestartet werden
4	Batch konnte wegen Fehlern in der Batch-Konfiguration nicht gestartet werden
143	Batch wurde vom Benutzer abgebrochen.
144	Batch wurde durch die Überschreitung der konfigurierten maximalen Laufzeit abgebrochen.

Tabelle 7: Allgemeine Return-Codes des Batchrahmens

## 5.10. Testmodus

Der Batch im Testmodus arbeitet analog zum normalen Wirkbetrieb, jedoch werden keine Änderungen an Datenbeständen vom Anwendungssystem oder Nachbarsystemen durchgeführt. Der Testmodus ist für den Betrieb wichtig, um Abschätzungen der Laufzeit durchführen zu können und somit den Batchbetrieb planen können. Weiter ist der Testlauf für Bereinigungsläufe wichtig, da so der Fachbereich sehen kann, welche Änderungen durch den Bereinigungslauf ausgeführt werden würden.

In diesem Kapitel werden Architekturmuster zur Umsetzung des Testmodus beschrieben:

- Der Batch arbeitet wie im Wirkbetrieb und statt der Commits findet immer ein Datenbank-Rollback statt (siehe Kapitel 5.10.1).

- Die Batch-Logik wird ausgeführt, jedoch finden keine Schreiboperationen in der Datenbank oder Aufrufe von Nachbarsystemen statt, welche Änderungen in deren Datenbestand zur Folge hätten (siehe Kapitel 5.10.2).

#### 5.10.1 Testmodus mit Rollback

Dieses Muster sieht vor, dass bei der Batchverarbeitung statt eines Commits ein Rollback ausgeführt wird, so dass die Änderungen, die durch den Batch erzeugt werden, nicht in die Datenbank geschrieben werden. Das Muster ist für datenbankorientierte Batches gut geeignet, die keine Änderungen an Nachbarsystemen erfordern.

Einige Batches vermerken ihren Arbeitsfortschritt in der Datenbank. Damit der Batch trotz Rollback nicht in eine Endlosschleife gerät, darf diese Änderung nicht zurückgerollt werden. Dies kann wie folgt umgesetzt werden:

Der Batch schreibt zu Beginn die IDs aller zu verarbeitenden Sätze in eine gesonderte „Task“-Tabelle. In jedem Schritt ermittelt der Batch einen Satz aus der Task-Tabelle, löscht diesen und verarbeitet den zugehörigen Datensatz. Im Testmodus wird das Löschen des Tasks in einer separaten Transaktion durchgeführt und so nicht zurückgerollt.

Folgendes Code-Beispiel demonstriert dieses Muster:

```
public VerarbeitungsErgebnis verarbeiteSatz() throws
BatchAusfuehrungsException {
    MeinBatchTask task =
meinBatchTaskDao.leseEintrag();
    meinBatchTaskDao.loesche(task);

    // Wenn Testmodus, neue Transaktion starten
    TransactionStatus txStatus = null;
    if (testmodus) {
        txStatus =
            transactionManager.getTransaction(new
DefaultTransactionDefinition(
TransactionDefinition.PROPROPAGATION_REQUIRES_NEW));
        txStatus.setRollbackOnly();
    }

    fristenkontrolle.pruefeFrist(task.getSatznummer());

    // Wenn Testmodus, Transaktion zurücksetzen
    if (testmodus) {
        transactionManager.rollback(txStatus);
    }
}
```

Die Umsetzung des Testmodus mit einem Datenbank-Rollback eignet sich vor allem zur Überprüfung der fehlerfreien Durchführung und bei der Bestimmung der Laufzeit eines Batches. Zusätzlich ist bei einer entsprechenden Protokollierung nachvollziehbar, welche Datensätze der Batch verarbeitet hat.

#### 5.10.2 Testmodus ohne Schreiboperationen

Ein weiteres Konzept für die Umsetzung des Testmodus sieht vor, dass ändernde Operationen in der Datenbank unterbunden werden. Änderungen



in der Datenbank oder in Nachbarsystemen werden durch entsprechende If-Abfragen abgefangen.

Dieses Muster ist dann einzusetzen, wenn die Realisierung durch einen Rollback nicht möglich oder angemessen ist. Durch Tests muss sichergestellt werden, dass nicht trotz Testmodus versehentlich Änderungen durchgeführt werden.

Das Muster ist geeignet, um im Testmodus durch die Auswertung der Batch-Protokolle und Logs die Auswirkungen bzw. durchgeführten Änderungen eines Batches vorab zu überprüfen. Es ist nicht geeignet zur Bestimmung der Laufzeit oder für die vollständige Sicherstellung der fehlerfreien Batchausführung, da Fehler bei Schreiboperationen in die Datenbank oder dem Aufruf von Nachbarsystemen nicht auftreten können.

## **6. Die Ausführungsbeans**

Im vorangegangenen Kapitel wurden die Eigenschaften des Batchrahmens erläutert. Dieser existiert bereits und muss verwendet werden, indem Ausführungsbeans dafür entwickelt werden. In diesem Kapitel werden Vorgaben für die Ausführungsbeans definiert.

### **6.1. Keine Transaktionssteuerung in einer Ausführungsbean**

Eine Ausführungsbean darf keine Transaktionen starten oder beenden. Sämtliche vom Batchrahmen aufgerufenen Operationen (Operationen von Interface BatchAusuehrungsBean) werden innerhalb einer Transaktion aufgerufen. Die Ausführungsbean muss sich hiermit nicht befassen.

### **6.2. Logging, Protokollierung und Statistik-Aufrufe implementieren**

Der Batchrahmen führt Logging nur im Fehlerfall durch. Die restlichen Informationen müssen durch die Ausführungsbean geloggt, protokolliert oder einer Statistik-Komponente übergeben.

Dazu wird die Ausführungsbean bei allen wichtigen Ereignissen aufgerufen:

- Beim Start des Batches
- Beim Schreiben eines Checkpoints
- Beim Beenden des Batches
- Bei der Verarbeitung eines Satzes

### **6.3. Fachliche Logik in den Komponenten der Fachanwendung implementieren**

Falls für die Verarbeitung im Batch Fachlogik benötigt wird, welche für die Webanwendung nicht benötigt wird, ist diese trotzdem den Fachanwendungskomponenten hinzuzufügen. Die Ausführungsbean erhält die Referenzen auf die Komponenten über Dependency Injection und ruft die Fachlogik dort auf.

Auch wenn in Sonderfällen Datenbank-Aufrufe direkt durch die Ausführungsbean ausgeführt werden müssen, ist die sonstige fachliche Logik an die Fachkomponenten der Fachanwendung zu delegieren.

### **6.4. Plausibilitätsprüfung in der Initialisierung**

Im Rahmen der Initialisierung hat die Ausführungsbean unter anderem die Aufgabe, die Konsistenz und Korrektheit der Eingabedaten zu prüfen. Dies kann beispielsweise ein erstes Durchlaufen der zu verarbeitenden Datei beinhalten. Werden hierbei Fehler erkannt, muss ein entsprechender Fehler geworfen werden.

### **6.5. In Initialisierung Schlüssel lesen, Satz-Verarbeitung über Lookups**

Falls die zu verarbeitenden Sätze eines Batches das Ergebnis einer Datenbank-Query sind, ist folgendermaßen vorzugehen:

- In Rahmen der Initialisierung ist die Query über eine Fachkomponente abzusetzen. Diese Query soll die (fachlichen) Schlüssel von Entitäten, nicht die Entitäten selbst auslesen.
- Die zurückgegebenen Schlüssel sind in einer Liste zu speichern und die Query ist zu schließen.
- Beim Aufruf für eine Satzverarbeitung ist die Entität über ihren Schlüssel aus der Datenbank auszulesen und die Verarbeitung durchzuführen.

Dies bietet gegenüber dem Auslesen von Entitäten in der Query folgende Vorteile:

- Würden Entitäten ausgelesen, wären diese nach einem Commit während der Verarbeitung nicht mehr mit einer Transaktion verbunden.

Hibernate liest Entitäten bereits bei der `hasNext()`-Abfrage eines Resultset-Iterators ab. So kommt es bei Checkpoints zwangsläufig zu toten Entitäten.<sup>5</sup>

- Durch das Ablegen der Schlüssel in einer Liste ist die Gesamtanzahl der Datensätze bekannt.

## 6.6. Informationen zum Batch-Benutzer bereitstellen

Batches müssen beim Start einen Benutzer authentifizieren und autorisieren, bevor der fachliche Teil der Batchverarbeitung starten kann. Der Batchrahmen fordert von der Ausführungsbean die Bereitstellung der Informationen zum Benutzer, um diesen über die T-Komponente Sicherheit zu authentifizieren. Dazu ist für das Ausführungsbean die Methode `getAuthenticationCredentials()` umzusetzen.

## 6.7. Fehlerbehandlung in Ausführungsbeans durchführen

Die Batches sind möglichst robust zu konstruieren: Falls auf ein fachliches Problem in der Ausführungsbean reagiert werden kann, sollte dies getan werden.

Der Batchrahmen unterstützt beispielsweise *nicht* das Auslassen von Datensätzen im Fehlerfall (etwa für eine Verarbeitung im nächsten Batch). Falls dies umgesetzt werden soll, ist eine entsprechende Verarbeitung in der Ausführungsbean zu implementieren.

Wenn eine Ausführungsbean einen Fehler wirft, so muss dies eine „`BeanAusfuehrungsException`“ oder ein davon erbender Fehler sein. In diesen Exceptions ist es möglich, den Return-Code des Batches zu definieren. Die Return-Codes sind für den konkreten Batch zu konfigurieren und müssen den Vorgaben in Abschnitt 5.9 entsprechen.

Wichtig ist, dass ein Batch bei einem Fehler, den der Batch nicht behandeln kann, abbricht und nicht endlos weiter läuft. Dieses Vorgehen ermöglicht es

---

<sup>5</sup> Falls (in einem ungewöhnlichen Sonderfall) mit Resultset-Iteratoren gearbeitet werden muss, so sollte mit Hibernate `ScrollableResults` gearbeitet werden. Siehe Hibernate-Referenzdokumentation.

dem Betrieb, die Ursache des Fehlers zu korrigieren und den Batch neu zu starten.

## 6.8. Beispiele Satzverarbeitung

Abschließend ist in die beispielhafte Satzverarbeitung in einer Batch-Ausführungsbean zu sehen. Die gezeigte Satzverarbeitung setzt den Testmodus um, die Anwendungslogik ist weiterhin im Anwendungskern umgesetzt und sie übernimmt Logging, Protokollierung und Statistik-Zählung.

```
@Override
public Verarbeitungsergebnis verarbeiteSatz() throws
BatchAusfuehrungsException {
    // Hole nächsten Task, wenn vorhanden.
    if (CollectionUtils.isEmpty(vorgangsIds)) {
        return new Verarbeitungsergebnis(null, true);
    }
    int vorgangsId = vorgangsIds.remove(vorgangsIds.size() - 1);

    // Wenn Testmodus, neue Transaktion starten
    TransactionStatus txStatus = pruefeStartSimulation();

    // Laden des Vorgangs und Aufruf der AWK Komponente
    // zum Versenden der Mitteilung
    VorgangRo vorgang =
        vorgangsverwaltung leseVorgang(vorgangsId);
    Antragsnummer antragsnummer = vorgang.getAntragsnummer();
    log.debug("Versende Erinnerung zur Anfrage für Antragsnummer: "
        + antragsnummer + ".");

    // Versende Erinnerung
    try {
        xxxBeteiligung.versendeErinnerungsnachricht(antragsnummer,
            vorgangsId);
        getBatchProtokoll().ergaenzeMeldung(
            new Verarbeitungsmeldung(String.valueOf(vorgangsId),
            antragsnummer.toString(),
            MeldungTyp.INFO, "Antragsnummer: " +
            antragsnummer));
        statistikErinnerungVersendet.erhoeheWert();
    } catch (AkteGesperrtException e) {
        getBatchProtokoll().ergaenzeMeldung(
            new Verarbeitungsmeldung(String.valueOf(vorgangsId),
            antragsnummer.toString(),
            MeldungTyp.WARNUNG, e.getFehlertext()));
    }

    // Wenn Testmodus, Transaktion zurücksetzen
    pruefeEndeSimulation(txStatus);

    return new Verarbeitungsergebnis(String.valueOf(vorgangsId),
        false);
}
```

### 6.8.1 Sonderfall Blocklöschung

Es ist nicht zwingend, dass in einem Satz auch nur eine Aktion (bspw. Löschung durchgeführt wird). Bei großen Datenmengen kann es durchaus Sinn machen, pro Satz eine bestimmte Anzahl Datensätze auf einmal zu verarbeiten, wie im folgenden Codebeispiel zu sehen ist. Die Variable blockgroesse ist dabei konfiguratativ zu setzen.

Wichtig ist in so einem Sonderfall, dass das Commit-Intervall des Batches entsprechend niedrig eingestellt ist (siehe Kapitel 5.2.1), da ansonsten  $\text{Blockgröße} \cdot \text{Commit-Intervall}$  Datensätze mit einer Transaktion verarbeitet werden.

```
@Override
public Verarbeitungsergebnis verarbeiteSatz() throws
BatchAusfuehrungsException {
    log.debug("Lösch Nachrichten älter als " + fristdatum + "
(Blockgröße " + blockgroesse + ").");

    // Wenn Testmodus, neue Transaktion starten
    TransactionStatus txStatus = pruefeStartSimulation();

    int anzahlGeloeschterEintraege =
nachrichtenverwaltung.loescheNachrichtenOhneAkte(fristdatum,
blockgroesse);
    log.debug(anzahlGeloeschterEintraege + " Nachrichten
gelöscht.");

    geloeschteNachrichtenStatistik.setWert(geloeschteNachrichtenStatist
ik.getWert()
        + anzahlGeloeschterEintraege);

    // Wenn Testmodus, Transaktion zurücksetzen
    pruefeEndeSimulation(txStatus);

    return new Verarbeitungsergebnis(null,
anzahlGeloeschterEintraege < blockgroesse);
}
```

### 6.8.2 Sonderfall eigenständige Transaktionssteuerung

Unter bestimmten Umständen kann es notwendig sein, die Transaktionssteuerung des Batchrahmens zu umgehen. Dies ist beispielsweise der Fall, wenn eine im Anwendungskern auftretende `DatenbankException` ignoriert werden soll. Das Auftreten einer solchen Exception führt dazu dass die Transaktion nicht mehr commitbar ist, selbst wenn die Exception gefangen und ignoriert wird.

Um dieser Problematik zu begegnen, kann in der Satzverarbeitung selber die Transaktionsklammer immer für einen Satz geöffnet und nach der Verarbeitung entweder committed oder zurückgerollt werden.

In diesem Fall sollte der Commit-Intervall des Batchrahmens entsprechend hoch konfiguriert werden, um unnötige Commits zu vermeiden, auch wenn in dieser Transaktion nichts passiert.

## 7. Ausnahmeregelungen

In bestimmten Situationen, z. B. bei der Verarbeitung extrem großer Datenmengen, kann der Fall auftreten, dass die Implementierung eines Batchschritts in Java nicht sinnvoll ist. In solchen Fällen muss dann eine Sonderlösung, wie z. B. ein SQL-Skript, gefunden werden.

Die Entscheidung, ob in einem solchen Einzelfall eine Sonderlösung gewählt werden kann, muss zwingend in Absprache mit dem Chefarchitekten bzw. Architekturboard erfolgen. Eine solche Abweichung von der Referenzarchitektur muss im Systementwurf in der Liste der Abweichungen aufgeführt und begründet werden. Dabei muss die Abwägung getroffen werden zwischen der Einhaltung der nicht-funktionalen Anforderungen und der Verringerung der Wartbarkeit durch mehrfach implementierte Funktionalität.

Wichtig ist, dass die einheitliche Außenschnittstelle der Batchschritte erhalten bleibt, d. h. anhand der übergebenen Parameter, Rückgabewerte, geloggten Daten darf für den Nutzer des Batchschritts nicht erkennbar sein, dass dieser in einer anderen Technologie umgesetzt ist.

Die Dokumentation der abweichenden Batchimplementierung erfolgt im Systementwurf und wird daraus in die Systemdokumentation übernommen.

## 8. Quellenverzeichnis

### [DatenzugriffDetailkonzept]

Detailkonzept Komponente Datenzugriff  
10\_Blaupausen\technische\_Architektur\Detailkonzept\_Komponente\_Datenzugriff.pdf.

### [SicherheitNutzerdok]

Nutzerdokumentation Sicherheit  
20\_Bausteine\Sicherheitskomponente\Nutzerdokumentation\_Sicherheit.pdf.

### [ÜberwachungKonfigKonzept]

Konzept Überwachung und Konfiguration  
20\_Bausteine\Ueberwachung\_Konfiguration\Konzept\_Ueberwachung-Konfiguration.pdf.

### [VorlageAnwendung]

Beispielimplementierung „Vorlage-Anwendung“  
Wird auf Anfrage bereitgestellt.

## 9. Abbildungsverzeichnis

Abbildung 1: Die Anwendungslogik eines Batches (Zielarchitektur).....	8
Abbildung 2: Ein Batch-Netz .....	9
Abbildung 3: Grobe Architektur des Batchrahmens .....	11
Abbildung 4: Bean-Zugriffe des Batchrahmens.....	16
Abbildung 5: Tabellen-Schema für die Batchrahmen-Tabelle .....	21
Abbildung 6: Die JMX Überwachung eines Batches .....	25



## 10. Tabellenverzeichnis

Tabelle 1: Kommandozeilen-Parameter des Batchrahmens .....	19
Tabelle 2: Die Properties des Batchrahmens .....	20
Tabelle 3: Das Schema der Tabelle BatchStatus .....	22
Tabelle 4: JMX-Attribute .....	26
Tabelle 5: Die Eigenschaften des Batchbenutzers .....	27
Tabelle 6: Format der Ergebnisdatei .....	30
Tabelle 7: Allgemeine Return-Codes des Batchrahmens .....	31