Bundesverwaltungsamt

IsyFact-Standard

Nutzerdokumentation Polling

Version 1.5
27.03.2015

IsyFact Standards

Ansprechpartner:

Referat Z II 2
Bundesverwaltungsamt
E-Mail: isyfact@bva.bund.de
Internet: www.isyfact.de

Dokumentinformationen

Dokumenten-ID:	Nutzerdokumentation_Polling.docx

Java Bibliothek / IT-System

Name	Art	Version
isy-polling	Bibliothek	siehe isyfact-bom v1.3.6

Inhaltsverzeichnis

1. Einleitung

Dieses Dokument beschreibt die Verwendung des Bausteins Polling.

In Fachanwendungen und Service-Gateways müssen manchmal Polling-basierte Schnittstellen angesprochen werden. Polling bedeutet, dass in regelmäßigen Intervallen neue Daten zur Verarbeitung abgeholt werden. Die Schnittstellen nutzen unterschiedliche technische Verfahren wie IMAP, Web-Services, HTTP-Invoker oder proprietäre Datenbank-basierte Schnittstellen, weitere sind denkbar.

Aus Gründen der Ausfallsicherheit soll die Abholung der Daten von mehreren Instanzen einer Anwendung durchgeführt werden. Diese Instanzen müssen synchronisiert werden, so dass Nachrichten nicht mehrfach verarbeitet werden. Die zugrunde liegenden Schnittstellen-Technologien bieten dafür kein Standardverfahren an.

In der folgenden Abbildung 1 wird diese Situation dargestellt.

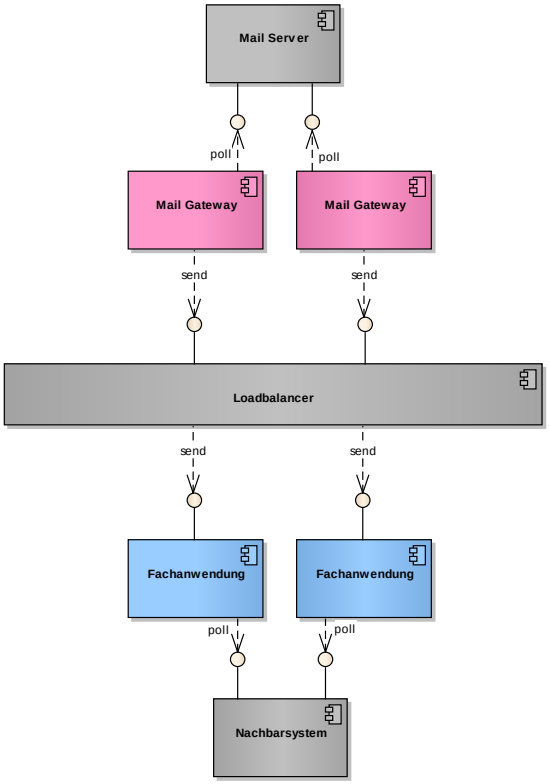

Abbildung 1: Polling unter Verwendung eines Loadbalancers

Der Baustein Polling definiert ein solches Verfahren und beschreibt die Bibliothek plis-polling, die dieses Verfahren implementiert.

Dieses Dokument richtet sich an Entwickler und Chefdesigner, die Polling-Funktionalität in eine Anwendung integrieren wollen.

2. Verfahren für das Polling

Das Polling erfolgt über mehrere Instanzen der Anwendung. Es darf zu einem Zeitpunkt aber nur eine Instanz das Polling aktiv durchführen, um das mehrfache Abrufen von Daten zu verhindern. Die Abstimmung darüber, welche Instanz das Polling aktiv durchführt, erfolgt automatisch.

Hierzu verwaltet jede Instanz den Zeitstempel ihrer letzten Polling-Aktivität und stellt den seit ihrer letzten Polling-Aktivität vergangenen Zeitraum per JMX bereit. Der Zeitraum wird anhand des Zeitstempels und der aktuellen Systemzeit beim Abruf des Wertes über JMX ermittelt. Es wird der Zeitraum und nicht der Zeitpunkt über JMX bereitgestellt, da hiermit das Problem nicht synchron laufender Uhren auf den einzelnen Servern keine Probleme bereitet.

Der Ablauf des Pollings in jeder Instanz ist dann wie folgt:

1. Es werden die seit der letzten Polling-Aktivität vergangenen Zeiträume aller anderen Instanzen über JMX ermittelt.

2. Wenn der Zeitraum seit der letzten Polling-Aktivität einer anderen Instanz kleiner als eine Wartezeit X ist, wird die Verarbeitung abgebrochen.

3. In allen anderen Fällen (keine Instanz erreichbar, keine Zeiträume gesetzt, alle Zeiträume größer als eine Wartezeit X) wird mit der Verarbeitung fortgefahren.

4. Zu Beginn des Abrufs von Nachrichten wird der eigene Zeitstempel auf den aktuellen Zeitpunkt gesetzt.

5. Während der Verarbeitung der einzelnen Nachrichten wird der Zeitstempel aktualisiert.

6. Am Ende der Verarbeitung aller abgerufenen Nachrichten wird der Zeitstempel nochmals aktualisiert.

In der folgenden Abbildung 2 wird das Verfahren dargestellt.

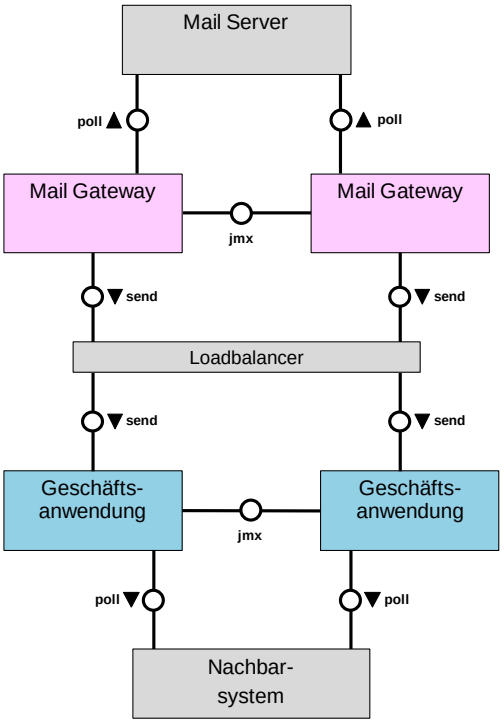

Abbildung 2: Angepasstes Verfahren zum Polling

Dieses Verfahren bietet im Regelfall einen sehr guten Schutz gegen Mehrfachverarbeitung. Es kann aber z.B. bei Netzproblemen dazu kommen, dass mehrere Instanzen parallel arbeiten. Dies lässt sich prinzipiell durch die Abstimmung per JMX nicht zu 100% ausschließen.

Daher ist es nötig, im Rahmen der Verarbeitung eingegangener Daten weitere Maßnahmen zu treffen. Hierzu wird in der Fachanwendung, die die Nachrichten verarbeitet, eine Tabelle für alle eingehenden Daten in der Datenbank angelegt.

Eine solche Tabelle ist häufig bereits vorhanden. In dieser Tabelle wird eine zusätzliche Spalte hinzugefügt, die einen Schlüssel aus dem Fremdsystem aufnimmt. Dieser Fremdsystemschlüssel kann je nach anlieferndem System z.B. die SMTP-Message-ID, ein Datenbankschlüssel oder auch ein Hashwert sein. Die Spalte wird mit einem Unique Constraint versehen. So können datenbankseitig Duplikate sicher erkannt werden.

Dafür ist es notwendig, diesen Fremdsystemschlüssel bis in die Fachanwendung zu transportieren, also z.B. vom Mail-Server über das Mail-Gateway. Da nur die Datenbank als transaktionales System die Eindeutigkeit der IDs garantieren kann, wird dies in Kauf genommen.

In der folgenden Abbildung 3 wird das noch einmal illustriert.

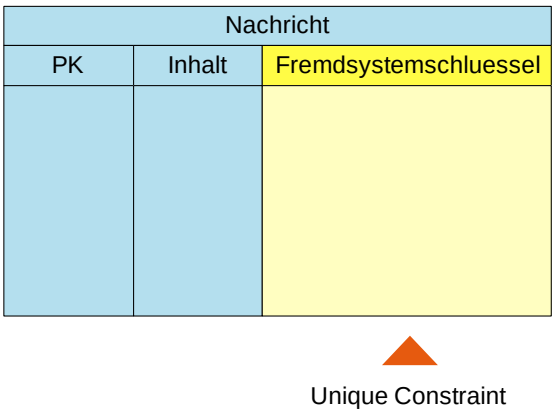

Abbildung 3: Nachrichten-Tabelle

3. Die Bibliothek plis-polling

Die IsyFact-Standard Bibliothek `plis-polling` implementiert den oben dargestellten Synchronisationsmechanismus über JMX. Die Bibliothek kann mit mehreren zu pollenden Datenquellen und beliebig vielen Instanzen der Anwendung zum Abfragen einer Datenquelle umgehen.

Zur weiteren Erläuterung der einzelnen Klassen der Bibliothek wird zunächst der Begriff „Polling-Cluster“ definiert:

> Ein **Polling-Cluster** besteht aus einer Menge von Anwendungsinstanzen, die jeweils die gleiche Datenquelle abfragen und wird durch eine innerhalb der Anwendung eindeutige ID identifiziert.

Ein Polling-Cluster wird durch die Klasse `PollingCluster` implementiert. Diese Klasse ist dafür zuständig, die Informationen eines Polling-Clusters zu verwalten. Hierzu zählen insbesondere der Zeitpunkt der letzten Polling-Aktivität und die Wartezeit. Die wesentlichen Parameter eines Polling-Clusters werden über die Anwendungskonfiguration festgelegt und beim Start der Anwendung eingelesen.

Die Verwaltung des Pollings für mehrere Nachrichtenquellen erfolgt über einen Polling-Verwalter (Klasse `PollingVerwalter`). Der Polling-Verwalter kennt und verwaltet alle PollingCluster der Anwendung. Die Klasse `PollingVerwalter` ist die einzige Klasse der Bibliothek, die eine Anwendungsimplementierung direkt nutzt.

Die Methode `startePolling` prüft, ob die aufrufende Anwendungsinstanz das Polling durchführen darf. Hierzu wird ihr die ID des Polling-Clusters als Argument übergeben. Sie liefert `true`, wenn das Polling gestartet werden darf, ansonsten `false`. Hierfür wird der seit der letzten Polling-Aktivität vergangene Zeitraum der anderen Instanzen des Polling-Clusters über JMX abgefragt und zusammen mit der im Polling-Cluster hinterlegten Wartezeit ausgewertet.

Die Methode `aktualisiereZeitpunktLetztePolling-Aktivitaet` setzt den Zeitpunkt der letzten Polling-Aktivität für den Polling-Cluster mit der übergebenen ID auf den aktuellen Systemzeitpunkt, die Methode `getZeitpunktLetztePollingAktivitaet` liefert den Zeitpunkt der letzten Polling-Aktivität für den Polling-Cluster mit der übergebenen ID.

Mit der Annotation `PollingAktion` ist es möglich, eine Methode zu annotieren, bei deren Aufruf dann der Zeitstempel für den angegebenen Polling-Cluster aktualisiert wird. Der zugehörige Interceptor ist in der Klasse `PollingAktionInterceptor` implementiert.

Schließlich ermöglicht die Klasse `PollingMBean` den anderen Instanzen, den seit der letzten Polling-Aktivität vergangenen Zeitraum über JMX abzufragen.

In der folgenden Abbildung 4 ist das Klassendiagramm der Bibliothek `plis-polling` dargestellt.

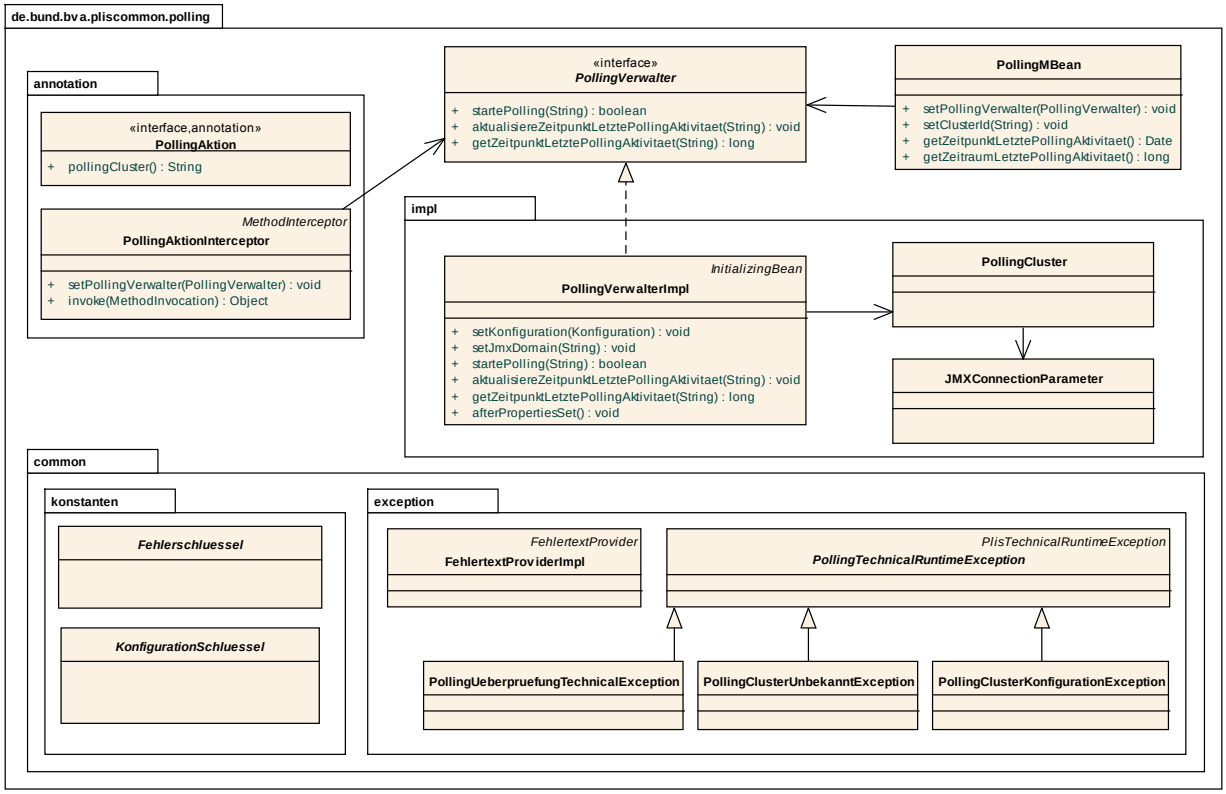

Abbildung 4: Klassendiagramm der Bibliothek `plis-polling`

4. Einbinden der Bibliothek in die Anwendung

Polling wird in der Regel in einem Verarbeitungszyklus implementiert, der über einen Timer gestartet wird. Der Verarbeitungszyklus kann eine oder nacheinander mehrere Datenquellen abfragen.

Das Einbinden der Bibliothek `plis-polling` in eine Anwendung erfolgt über Spring. JMX-Verbindungen und Polling-Cluster werden über die Anwendungskonfiguration definiert.

Im Folgenden wird im Detail beschrieben, wie die Bibliothek eingebunden und genutzt wird.

4.1. Anwendungskonfiguration

In der Anwendungskonfiguration werden Polling-Cluster und JMX-Verbindungsparameter definiert. Bei der Definition der Polling-Cluster hat man die Möglichkeit einen Cluster für einen gesamten Verarbeitungszyklus oder einen Cluster für jede abzufragende Datenquelle zu definieren. Welche der beiden Möglichkeiten gewählt wird, hängt vom konkreten Anwendungsfall ab und muss beim Entwurf der Anwendung entschieden werden.

4.1.1 Lastverteilung über Cluster-Definition

Wird ein Polling-Cluster je zu pollender Datenquelle festgelegt, so kann eine gewisse Lastverteilung erreicht werden wie folgendes Beispiel zeigt:

Beispiel:

> In einer Anwendung ist ein Verarbeitungszyklus so realisiert, dass drei E-Mail-Konten nacheinander abgefragt werden. Der Verarbeitungszyklus wird periodisch über einen Timer gestartet. Es gibt zwei Instanzen der Anwendung. Wird für jedes E-Mail-Konto ein eigener Polling-Cluster definiert, so ist der Ablauf der folgende:
>
> Der Timer im Knoten 1 startet den Verarbeitungszyklus. Es wird zunächst der Polling-Verwalter gefragt, ob das Polling für den Cluster 1 gestartet werden darf. Knoten 2 hat das Polling noch nicht begonnen, so dass der Polling-Verwalter im Knoten 1 die Verarbeitung der Nachrichten aus dem E-Mail-Konto 1 erlaubt.
>
> Zwischenzeitlich startet auch der Timer im Knoten 2 einen Verarbeitungszyklus. Der Polling-Verwalter im Knoten 2 erkennt, dass das Polling im Knoten 1 für den Cluster 1 bereits aktiv ist und verweigert die Erlaubnis, die Nachrichten des E-Mail-Konto 1 abzuarbeiten. Die Anwendung fragt den Polling-Verwalter daher als nächstes um Erlaubnis, das Polling für den Cluster 2 durchzuführen. Das wird erlaubt und Knoten 2 verarbeitet die Nachrichten des E-Mail-Konto 2.

Während Knoten 2 noch mit den Nachrichten des E-Mail-Konto 2 beschäftigt ist, hat Knoten 1 alle Nachrichten aus E-Mail-Konto 1 verarbeitet. Er fragt jetzt den Polling-Verwalter um Erlaubnis, das Polling für den Cluster 2 durchzuführen. Die Erlaubnis wird verweigert, weil Knoten 2 bereits damit beschäftigt ist. Daher fragt die Anwendung als nächstes um Erlaubnis, das Polling für den Cluster 3 durchzuführen. Das wird erlaubt und Knoten 1 verarbeitet die Nachrichten aus dem E-Mail-Konto 3.

Knoten 2 ist mit der Verarbeitung des E-Mail-Konto 2 fertig und möchte nun E-Mail-Konto 3 verarbeiten, was aber verweigert wird, da Knoten 1 bereits damit beschäftigt ist. Damit ist der Verarbeitungszyklus im Knoten 2 beendet.

Dieses Beispiel zeigt, dass E-Mail-Konten parallel von beiden Instanzen abgearbeitet werden können.

4.1.2 Konfigurationsklassen

Polling-Cluster und JMX-Verbindungsparameter werden über Konfigurationsklassen konfiguriert. Eine Konfigurationsklasse definiert zunächst eine Menge zusammengehöriger Konfigurationsparameter. Von jeder Konfigurationsklasse kann es mehrere Ausprägungen geben. Eine Konfigurationsklasse wird durch ein festes Namenspräfix der Konfigurationsparameter beschrieben.

Werden Ausprägungen von Konfigurationsklassen konfiguriert, so werden die IDs aller Ausprägungen unter einem Konfigurationsparameter aufgelistet. Konfigurationsparameter einer Ausprägung haben dann das Namensschema <Konfigurationsklassen-Präfix>.<ID der Ausprägung>.<Name des Konfigurationsparameters>.

Beispiel:

Die Konfigurationsklasse „JMX-Verbindung“ hat das Präfix `polling.jmxverbindung`.

Für die Konfigurationsklasse „JMX-Verbindung“ gibt es die Konfigurationsparameter `host`, `port`, `benutzer` und `passwort`.

Soll eine Ausprägung einer JMX-Verbindung konfiguriert werden, so sieht das dann wie folgt aus:

```
# Kommaseparierte Liste mit den Ids der Verbindungsparameter zu
# den übrigen Clusterservern
polling.jmxverbindung.ids = SERVER2

# Verbindungsparameter zum anderen Knoten
# Hostangabe für den JMX-Zugriff
polling.jmxverbindung.SERVER2.host = localhost
# Portangabe für den JMX-Zugriff
polling.jmxverbindung.SERVER2.port = 9010
# Benutzerkennung für den JMX-Zugriff
```

```
polling.jmxverbindung.SERVER2.benutzer = userid
# Kennwort für den JMX-Zugriff
polling.jmxverbindung.SERVER2.passwort = pwd
```

Wie man an dem Beispiel sieht, gilt die Konvention, dass IDs großgeschrieben werden. Damit kann an einzelnen Konfigurationseinträgen sofort erkannt werden, dass sie zu einer Konfigurationsklasse gehören.

Für das Polling gibt es die folgenden Konfigurationsklassen:

- **JMX-Verbindungen** beinhalten die JMX-Verbindungsparameter zu den anderen Knoten des Clusters. Sie werden über eine ID identifiziert. Das Präfix der Konfigurationseinträge für JMX-Parameter lautet `polling.jmxverbindung`.

- **Polling-Cluster** enthalten die Angaben zur Steuerung des Pollings. Ein Polling-Cluster wird über eine ID identifiziert. Das Präfix der Konfigurationseinträge für Polling-Cluster lautet `polling.cluster`.

4.1.3 Konfigurationsparameter

JMX-Verbindungen haben die folgenden Attribute:

`polling.jmxverbindung.ids`:
: Kommaseparierte Liste mit den IDs der Verbindungsparameter zu den übrigen Clusterservern.

`polling.jmxverbindung.<ID>.host`:
: Hostangabe für den JMX-Zugriff.

`polling.jmxverbindung.<ID>.port`:
: Portangabe für den JMX-Zugriff.

`polling.jmxverbindung.<ID>.benutzer`:
: Benutzerkennung für den JMX-Zugriff

`polling.jmxverbindung.<ID>.passwort`:
: Kennwort für den JMX-Zugriff

Polling-Cluster haben die folgenden Attribute:

`polling.cluster.ids`:
: Kommaseparierte Liste der Cluster-IDs.

`polling.cluster.<ID>.name`:
: Name des Polling-Clusters. Der hier festgelegte Name wird zur MBean-Identifikation benutzt und ist in der JMX-Konsole sichtbar.

`polling.cluster.<ID>.wartezeit`:
: Wartezeit in Sekunden, die abgelaufen sein muss, damit diese

Anwendung das Polling übernehmen kann. Dieser Wert sollte doppelt so groß sein, wie der Delay-Wert des Timers, der den Verarbeitungszyklus auslöst. Die Wartezeit muss mindestens 10 Sekunden betragen.

`polling.cluster.<ID>.jmxverbindungen`:

Kommaseparierte Liste von IDs der Verbindungsparameter zu den übrigen Clusterservern. Dieser Eintrag ist optional und wird in der Regel nicht benötigt. Wird er weggelassen, so werden alle für das Polling konfigurierten JMX-Verbindungen zugeordnet.

4.1.4 Konfiguration für den Test

Für Tests der Anwendung, insbesondere für lokale Entwicklertests, stehen in der Regel nicht mehrere Instanzen der Anwendung zur Verfügung. In diesem Fall kann das konfigurierte Polling die Tests behindern.

Für Tests kann der Polling-Verwalter in den Standalone-Modus versetzt werden. In diesem Modus erkennt der Polling-Verwalter, dass keine Cluster-Partner existieren und die Polling-Aktionen werden immer zugelassen.

Der Standalone-Modus wird automatisch gesetzt, wenn in der Konfiguration der Parameter `polling.jmxverbindung.ids` weggelassen bzw. auskommentiert oder der Wert leer gelassen wird. Da dieses Verhalten in der Regel im Produktivbetrieb nicht erwünscht ist, wird die folgende Warnung in die Log-Ausgabe geschrieben:

Für das Polling der Anwendung wurden keine JMX-Verbindungsparameter angegeben! Der Polling-Modus wurde auf "Standalone" gesetzt!

4.1.5 Beispiel für eine Polling-Konfiguration

Im Folgenden ist eine vollständige Polling-Konfiguration für einen Cluster aufgeführt, der aus insgesamt zwei Instanzen der Anwendung besteht.

```
# ------------------------------------------------------------
# Parameter für das Polling
# ------------------------------------------------------------
# Kommaseparierte Liste mit den Ids der Verbindungsparameter
# zu den übrigen Clusterservern
polling.jmxverbindung.ids = SERVER2

# Verbindungsparameter zum anderen Knoten
# Hostangabe für den JMX-Zugriff
polling.jmxverbindung.SERVER2.host = localhost
# Portangabe für den JMX-Zugriff
polling.jmxverbindung.SERVER2.port = 9010
# Benutzerkennung für den JMX-Zugriff
polling.jmxverbindung.SERVER2.benutzer = userid
# Kennwort für den JMX-Zugriff
polling.jmxverbindung.SERVER2.passwort = pwd


# Kommaseparierte Liste der Cluster-Ids, zu denen die
# Anwendung gehört
polling.cluster.ids = MAILABRUF_CLUSTER
```

```
# Name des Clusters. Dieser Name wird zur Bildung der MBean-
# Identifikation verwendet.
polling.cluster.MAILABRUF_CLUSTER.name = XY-Nachrichten
# Wartezeit in Sekunden, die abgelaufen sein muss, damit
# diese Anwendung das Polling übernehmen kann.
polling.cluster.MAILABRUF_CLUSTER.wartezeit = 600
```

Cluster können nicht dynamisch nur durch die Konfiguration erzeugt werden. Sie sind vielmehr eng mit der Anwendungslogik verknüpft. Daher ist es sinnvoll, nur die Konfigurationsparameter in die betriebliche Property-Datei der Anwendung zu legen, die sich auf die Verbindungsparameter beziehen.

4.1.6 Beispiel für eine Polling-Konfiguration mit Lastverteilung

Im Folgenden ist die Polling-Konfiguration für zwei zu pollende Datenquellen mit jeweils eigenem Polling-Cluster aufgeführt. Die beiden Cluster bestehen jeweils aus den zwei gleichen Instanzen der Anwendung. Wie in Kapitel 4.1.1 beschrieben, kann so eine Lastverteilung erfolgen.

```
# ------------------------------------------------------------
# Parameter für das Polling
# ------------------------------------------------------------
# Kommaseparierte Liste mit den Ids der Verbindungsparameter
# zu den übrigen Clusterservern
polling.jmxverbindung.ids = SERVER2

# Verbindungsparameter zum anderen Knoten
# Hostangabe für den JMX-Zugriff
polling.jmxverbindung.SERVER2.host = localhost
# Portangabe für den JMX-Zugriff
polling.jmxverbindung.SERVER2.port = 9010
# Benutzerkennung für den JMX-Zugriff
polling.jmxverbindung.SERVER2.benutzer = userid
# Kennwort für den JMX-Zugriff
polling.jmxverbindung.SERVER2.passwort = pwd

# Kommaseparierte Liste der Cluster-Ids, zu denen die
# Anwendung gehört
polling.cluster.ids = POSTFACH1_CLUSTER,POSTFACH2_CLUSTER

# Parameter des POSTFACH1_CLUSTER
# Name des Clusters. Dieser Name wird zur Bildung der MBean-
# Identifikation verwendet.
polling.cluster.POSTFACH1_CLUSTER.name = Postfachabruf-1
# Wartezeit in Sekunden, die abgelaufen sein muss, damit
# diese Anwendung das Polling übernehmen kann.
polling.cluster.POSTFACH1_CLUSTER.wartezeit = 600

# Parameter des POSTFACH2_CLUSTER
# Name des Clusters. Dieser Name wird zur Bildung der MBean-
# Identifikation verwendet.
polling.cluster.POSTFACH2_CLUSTER.name = Postfachabruf-2
# Wartezeit in Sekunden, die abgelaufen sein muss, damit
# diese Anwendung das Polling übernehmen kann.
polling.cluster.POSTFACH2_CLUSTER.wartezeit = 600
```

4.2. Spring-Konfiguration

Über Spring werden mindestens der Polling-Verwalter und je Cluster eine MBean konfiguriert. Wird die PollingAktion-Annotation verwendet, muss der entsprechende Interceptor noch konfiguriert werden.

Im Folgenden werden Beispiele für die Spring-Konfiguration aufgeführt.

Beispiel: Konfiguration des Polling-Verwalters

```
<!-- Polling-Verwalter -->
<bean id="pollingVerwalter" class="de.bund.bva.pliscommon.polling.impl.PollingVerwalterImpl">
    <property name="konfiguration" ref="konfiguration" />
    <property name="jmxDomain" value="de.bund.bva.domaene.anwendung" />
</bean>
```

Die Property `konfiguration` enthält die Referenz auf die Komponente „Konfiguration“, die Property `jmxDomain` enthält den Namen der JMX-Domain. Das ist in der Regel das Basis-Package der Anwendung.

Beispiel: Konfiguration der MBeans

```
<bean id="mailabrufClusterMonitor" class="de.bund.bva.pliscommon.polling.PollingMBean"
      depends-on="mBeanExporter">
    <property name="pollingVerwalter" ref="pollingVerwalter" />
    <property name="clusterId" value="MAILABRUF_CLUSTER" />
</bean>
```

Die Property `pollingVerwalter` enthält die Referenz auf die Komponente „Polling-Verwalter“, die Property `clusterId` enthält die ID des Polling-Clusters, für den sie den seit der letzten Polling-Aktivität vergangenen Zeitraum liefern soll.

Die Einbindung in JMX erfolgt über den MBean-Exporter dann wie folgt:

```
<bean id="mBeanExporter" class = "org.springframework.jmx.export.MBeanExporter">
```

```
    <property name="assembler">
        <bean class = "org.springframework.jmx.export.assembler.MetadataMBeanInfoAssembler">
            <property name="attributeSource">
                <bean class = "org.springframework.jmx.export.annotation.AnnotationJmxAttributeSource" />
            </property>
        </bean>
    </property>
    <property name="autodetect" value="false" />
    <property name="beans">
        <map>
            <entry key=
         "de.bund.bva.domaene.anwendung:type=PollingStatus,name=Polling-Aktivitaet-${polling.cluster.MAILABRUF_CLUSTER.name}"
                    value-ref="mailabrufClusterMonitor" />
        </map>
    </property>
</bean>
```

Hierbei ist zu beachten, dass der Cluster-Name aus der Konfiguration hier für die Bildung des Keys für die MBeans verwendet wird.

Beispiel: Konfiguration des Polling-Aktion-Interceptors

```
<!-- @PollingAktion Annotation einschalten -->
<bean id="pollingAktionInterceptor" class="de.bund.bva.pliscommon.polling.annotation.PollingAktionInterceptor">
    <property name="pollingVerwalter" ref="pollingVerwalter" />
</bean>

<aop:config>
    <aop:pointcut id="pollingAktionPointcut"
                  expression="@annotation(de.bund.bva.pliscommon.polling.annotation.PollingAktion) ||
@within(de.bund.bva.pliscommon.polling.annotation.PollingAktion)" />
    <aop:advisor pointcut-ref="pollingAktionPointcut"
                 advice-ref="pollingAktionInterceptor" />
</aop:config>
```

4.3. Nutzung im Code

Wie bereits erwähnt, wird Polling in der Regel in einem Verarbeitungszyklus implementiert, der über einen Timer gestartet wird. Der Verarbeitungszyklus kann eine oder nacheinander mehrere Datenquellen abfragen, für die jeweils ein Polling-Cluster definiert sein kann. Für jeden Polling-Cluster werden in einem Verarbeitungszyklus die folgenden Schritte ausgeführt:

1. Abfrage des Polling-Verwalters, ob das Polling für den Cluster gestartet werden darf. Ist das nicht der Fall, ist die Verarbeitung für den Cluster beendet.

```
if (!pollingVerwalter.startePolling("MAILABRUF_CLUSTER")) {
    LOG.info("Verarbeitung wurde nicht gestartet, da die " +
             "Wartezeit für den Cluster mit der ID " +
             "\"MAILABRUF_CLUSTER\" noch nicht " +
             "abgelaufen ist.");
    return;
}
fuehreVerarbeitungDurch();
```

2. Durchführen der Verarbeitung für jeden Datensatz. Am Ende der Verarbeitung eines Datensatzes wird der Zeitpunkt der letzten Polling-Aktivität aktualisiert.

```
// ...
for (Datensatz datensatz: datensaetze) {
    // Verarbeite Datensatz
    // ...
    pollingVerwalter.
        aktualisiereZeitpunktLetztePollingAktivitaet(
            "MAILABRUF_CLUSTER");
}
// ...
```

oder

```
/**
 * Führt eine Polling-Aktion aus
 */
@PollingAktion(pollingCluster="MAILABRUF_CLUSTER")
public void doPollingAktion () {
    // Verarbeite Datensatz
    // ...
}

// ...
    for (Datensatz datensatz: datensaetze) {
        // Verarbeite Datensatz
        // ...
        doPollingAktion();
    }
// ...
```

5. Abkürzungsverzeichnis

BVA	Bundesverwaltungsamt
HTTP	Hypertext Transfer Protocol: Netzwerkprotokoll zur Übertragung von Daten.
IMAP	Internet Message Access Protocol: Netzwerkprotokoll für den Abruf von E-Mails.
JMX	Java Management Extensions: Spezifikation zur Verwaltung und Überwachung von Java-Anwendungen.
MBean	Managed Bean: Java-Objekte zur und Bereitstellung von Eigenschaften über JMX.
SMTP	Simple Mail Transfer Protocol: Netzwerkprotokoll für den Austausch von E-Mails.

6. Abbildungsverzeichnis