Meldungs-management

Aufgabe

In allen Ebenen einer Software-Applikation gibt es Nachrichten und Fehlermeldungen. Das sind aktiv und unaufgefordert ausgesendete Meldungen der Software-Anwendung an einen Nutzer oder ein technische Komponente, die eine Reaktion bei Erhalt einer Meldung ausführen. Sie haben unterschiedlichen Inhalt und unterschiedliche Semantik und können in allen Ebenen der Applikation entstehen. In CBA werden Nachrichten und Fehlermeldungen unter dem Begriff der Meldungen zusammengefasst und einheitlich behandelt.

Fehlerbehandlung, Logging und Monitoring stellen spezielle Reaktionen auf Meldungen dar, wobei ein und dieselbe Meldung mehrfache Reaktionen auslösen kann. Während normale Meldungen in der Regel an der Bedienoberfläche dem interaktiven Nutzer angezeigt werden, werden Logging-Informationen in eine Logging-Datenbank gespeichert, um von einem Administrator bei Bedarf eingesehen zu werden. Ein Monitoring-System wertet einen bestimmten Teil der Meldungen aus, um einen permanenten Überblick über die Arbeit des Systems zu erhalten. Meldungen können auch Fragen an den Nutzer darstellen oder einen Workflow initiieren.

Eine Meldung trägt in CBA neben der eigentlichen Information einen Zeitstempel und häufig Zusatzinformationen wie Sitzung, Transaktion, Nutzer etc. Damit sind aus Meldungen bei gezielter Nutzung z.B. auch Aufrufhäufigkeiten, Aufrufdauer u.a. ermittelbar. Das Ausbleiben von Meldungen kann auch ausgewertet werden.

Das CBA-Meldungs-Management ist mandanten- und mehrsprachfähig und kann parametrierte Meldungen generieren. Diese kann neben Fehlern auch jede andere Art von Meldungen und Textbausteinen verwalten. Das können z.B. Abfragen für Bedienoberflächen, formatierte Ausschriften, Textbausteine für E-Mails oder Protokoll-Einträge sein.

Anforderung

CBA implementiert ein einheitliches Meldungs-Management, das auch die Anforderungen an die Fehlerbehandlung, das Logging und das Monitoring berücksichtigt:

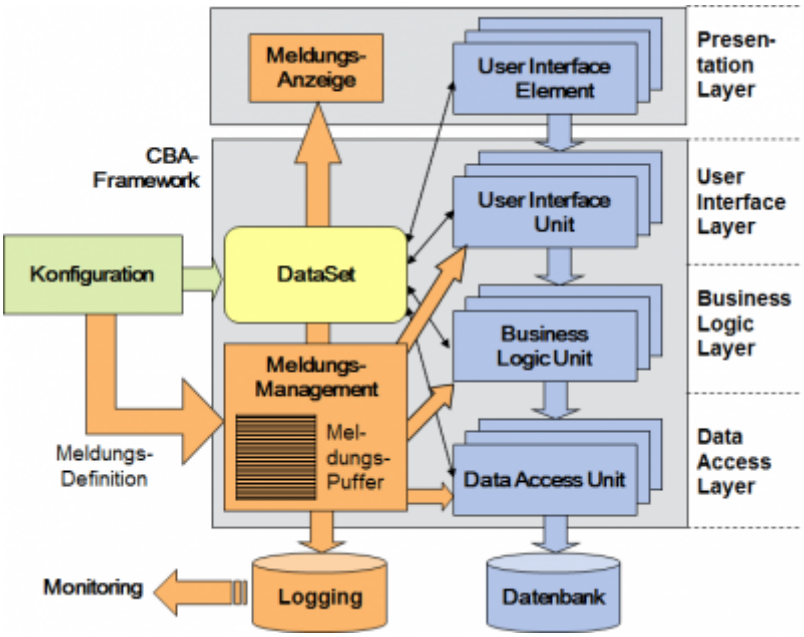
- Meldungen werden nach ihrer Bedeutung klassifiziert.
- Meldungen definieren zusammen mit Systemeinstellungen eindeutig die notwendigen Reaktionen. Die Reaktion auf Meldungen ist parametrierbar.
- Beim Senden einer Meldung (z.B. eines Fehlers) kann wahlweise die ausgeführte Funktion durch einen CbaException abgebrochen oder innerhalb einer logischen Transaktionsklammer (z.B. innerhalb eines Service-Aufrufes) alle Meldungen gesammelt und komplett übergeben werden. Damit ist eine fehlertolerante Arbeitsweise möglich.
- Die Meldungen können einen oder mehrere Parameter enthalten, die durch die Funktionen übergeben werden, z.B. den Dateinamen einer nicht gefundenen Datei.
- Das Meldungs-Management generiert aus den parametrierten Meldungen einen lesbaren Meldungstext.
- Neben Fehlern können auch andere Arten von Meldungen und Textbausteinen verwaltet werden

wie z.B. Abfragen für Bedienoberflächen oder formatierte Ausschriften.
- Meldungen untergelagerter Programmsysteme wie z.B. Datenbank-Systeme oder Datei-Systeme werden in das einheitliche Meldungs-Management überführt.
- Das CBA-Meldungs-Management ist mehrsprachfähig, d.h. die Meldungen werden in der Sprache des Nutzers generiert.
- In verteilten Anwendungen sind Meldungen transparent, d.h. sie werden so zwischen Service-Aufruf und Service kommuniziert, das ein einheitliches Verhalten unabhängig vom Ort der Meldungs-Entstehung realisiert wird.

Architektur

Das Meldungs-Management ist in das CBA-Framework eingebettet. Es besteht aus

- der Meldungs-Definition
- dem temporären Meldungs-Puffer
- den Meldungs-Funktionen
- vorgefertigten Elementen zur Meldungs-Anzeige
- der Logging-Datenbank
- der Möglichkeit der Anbindung eines Monitoring-Systems



Konfiguration

Die potenziellen Meldungen werden mit einer Meldungs-Nummer konfiguriert. In der Meldungs-Definition sind Platzhalter für die später einzufügenden Parameter definierbar. Die Meldungstexte sind mehrsprachfähig.

Funktionen

Das CBA-Framework stellt eine Menge von Funktionen zum Meldungs-Management zur Verfügung:

- **clearMessage** Löschen des Meldungs-Puffers
- **setMessage** Einfügen einer Meldung in den Meldungs-Puffer
- **getMessageText** Konvertierung einer Meldung in einen Text
- **getMessages** Rückgabe des Meldungs-Puffers
- **saveMessage** Speichern der Meldungen in der Logging-Datenbank
- **removeMessage** Löschen der zuletzt eingestellten Meldung aus dem Meldungs-Puffer

Der Funktion **setMessage** werden eine Meldungsnummer, ggf. Meldungs-Parameter (z.B. Informationen über das fehlerhafte Objekt u.ä.), die Klassifikation und ein Flag zur Verhaltens-Spezifikation übergeben. Weitere Parameter können die Meldung stärker spezifizieren und sind insbesondere für einen Fehleranalyse wichtig. Mit Hilfe der Funktion getMessageText wird anhand der Meldungsnummer eine sprachabhängige Zeichenkette ermittelt, die mit den angegebenen Meldungs-Parametern formatiert wird. {0} ist dabei der Platzhalter für den ersten Meldungs-Parameter, {1} für den zweiten etc. Die Meldungs-Parameter sind eine Zeichenkette, in der beliebig viele Parameter durch â€š|â€™ getrennt hintereinander angegeben werden können. Automatisch wird der Meldung der Zeitstempel, die Session-Zuordnung und (wenn vorhanden) die Transaktionsnummer angefügt, sodass Meldungen in einen Kontext eingeordnet werden können.

Beispiel:

- **Meldungsnummer**: 1234
- **Meldungstext**: 'Die Datei '{0}' hat folgenden Fehler verursacht: {1}
- **Aufruf**: setMessage (1234, „bild.gif|Datei schreibgeschützt“, „E“, „e6“, myTable, null, 4711);
- **Meldung**: Die Datei 'bild.gif' hat folgenden Fehler verursacht: Datei schreibgeschützt

Die Funktion **removeMessage** kann dazu genutzt werden, die zuletzt eingestellte Meldung aus dem Meldungs-Puffer zu löschen. Damit können programmtechnisch Fehler abgefangen und die Auswirkungen einer Meldung rückgängig gemachtwerden.

Die Funktion **getMessageText** kann auch aufgerufen werden, ohne dass eine Meldung in den Meldungs-Puffer geschrieben wird. In den Bedienoberflächen können so z.B. mehrsprachige Texte für Abfragen, Meldungen u.ä. ermittelt werden.

getMessages liest den Meldungs-Puffer aus, um alle Meldungen der Transaktion auswerten zu können.

Mit **saveMessage** werden die relevanten Meldungen zu Zwecken des Logging und des Monitoring in die Datenbank abgespeichert. Das Logging umfasst die Meldungen, bei denen das Flag 'l' gesetzt ist und deren Level kleiner oder gleich dem im System eingestellten Logging-Level ist. Sie können u.a. mit System-Werkzeugen aus der Datenbank ausgelesen werden. Das **Monitoring** erfolgt mit einer eigenen Anwendung, die die in der Datenbank abgespeicherten Logging-Informationen nach einer eigenen Logik auswertet. über die konfigurierte BLU des Meldungs-Puffers kann die Monitoring-Anwendung über das Vorhandensein neuer Logging-Informationen informiert werden und damit zeitnah reagieren.

Meldungs-Klassifikation

Meldungen unterliegen in CBA folgender Klassifikation:

F .. fatale Fehler	Fehler, die die Weiterarbeit der Applikation verhindern oder wesentlich beeinträchtigen
E .. Fehler	Fehler, die in der Arbeit einer Applikation auftreten können, nur lokale Auswirkungen haben und auf die der Anwender reagieren kann
W .. Warnungen	Hinweise des Systems auf Inkonsistenzen, ungünstige Zustände und ähnliches
I .. Informationen	Informationen an den Nutzer
Q .. Fragen	Fragen an den Nutzer, die mit Ja oder Nein beantwortet werden können

Die Klassifikation wird beim Aufruf der Funktion **setMessage** übergeben.

Das Verhalten eines **setMessage**-Aufrufes wird durch das Flag bestimmt. Es kann folgende Werte beinhalten:

e .. Exception	Die Meldung löst nach ihrer Behandlung einen CbaException aus, der in einer beliebigen Aufruf-Hierarchie-Ebene abgefangen werden kann.
1 .. Logging-Level 1	Logging- und Monitoring-Informationen für fatale Systemfehler
2 .. Logging-Level 2	Logging- und Monitoring-Informationen für fatale Konfigurationsfehler
3 .. Logging-Level 3	Logging- und Monitoring-Informationen für fatale Anwendungsfehler
4 .. Logging-Level 4	Logging- und Monitoring-Informationen für Systemfehler
5 .. Logging-Level 5	Logging- und Monitoring-Informationen für Konfigurationsfehler
6 .. Logging-Level 6	Logging- und Monitoring-Informationen für Anwendungsfehler
7 .. Logging-Level 7	anwendungsspezifische Logging- und Monitoring-Informationen
8 .. Logging-Level 8	anwendungsspezifische Logging- und Monitoring-Informationen
9 .. Logging-Level 9	anwendungsspezifische Logging- und Monitoring-Informationen

Die Meldungen werden an das Logging und Monitoring weitergeleitet, wenn das angegebene Level kleiner oder gleich dem beim **saveMessage**-Aufruf angegebenen Level ist.

Verhalten

Der Meldungs-Puffer wird zu Transaktions-Beginn automatisch durch Aufruf der Funktion **clearMessage** gelehrt. Der Transaktions-Beginn ist implizit durch einen Service-Aufruf gegeben, kann aber auch explizit durch Aufruf der Funktion **BlTransactionBegin** durch den Anwender ausgelöst werden. Es werden alle Meldungen der Transaktion im Meldungs-Puffer gesammelt, bis die nächste Transaktion begonnen hat. Damit kann auch nach Abschluss der Transaktion mit Hilfe der Funktion **getMessages** die Liste aller bei der letzten Transaktion aufgelaufenen Meldungen zurückgeliefert werden.

Durch die gezielte Steuerung der Exceptions ist das Aufsammeln von Meldungen möglich, ohne dass der Programmablauf unterbrochen wird. Es können fehlertolerante Services implementiert werden, die z.B. fehlerfreie Informationen bearbeiten und fehlerbehaftete zurückweisen. Mit Hilfe der Meldungen sind die zurückgewiesenen Informationen und der Rückweisungsgrund eindeutig nachvollziehbar.

In verteilten Anwendungen sorgt das CBA-Framework dafür, dass die Meldungen des aufgerufenen Services in die Meldungsliste der aufrufenden Anwendung überführt und ggf. ein Exception ausgelöst wird. Damit ist das Verhalten über Domänengrenzen hinweg einheitlich.

System Meldungen

System-Meldungen werden automatisch bei folgenden Ereignissen durch das CBA-Framework generiert:

- im CBA-Framework erkannte Fehler
- Fehler untergelagerter Systeme (z.B. Datei-Verwaltung, Datenbank, ...)
- Beginn und Ende eines Service-Aufrufes
- Instanziierung einer Unit

Alle System-Meldungen des CBA-Framework werden über die Funktion setMessage generiert.

Anwendung

Meldungen werden im Programm-Code an verschiedenen Stellen generiert. Während einer Transaktion werden alle Meldungen im Meldungs-Puffer zwischengespeichert, sodass auch eine Folge von Meldungen erzeugt werden kann. Ein wichtiges Anwendungs-Beispiel dafür sind Validierungen. Die Validierung wird nicht beim ersten erkannten Fehler beendet, sondern wird fortgeführt. Dadurch kann ein besserer Bedienkomfort erreicht werden, weil in einem Zyklus mehrere Fehler beseitigt werden können. Durch eine gezielte Steuerung der Exceptions ist das Aufsammeln von Meldungen möglich, ohne dass der Programmablauf unterbrochen wird.

Nachfolgend ein Beispiel für die Anwendung der Meldungs-Funktionen:

```
      string MyFunction (...)
        {
            string result = null;
            try
            {
                // ... gewünschte Funktionalität

                // spezieller Fehler
                if (...) setMessage(1001, [par0]+"|"+[par1], "E", "e", null,
null, 0);
                   // [par0] würde den Platzhalter {0} und [par1] den
Platzhalter {1} ersetzen
        // Flag  'e' => nach Fehlerbehandlung wird ein CbaException
ausgelöst

                // ... weitere Funktionalität

                // ... Rückgabe einer sprachabhängigen Meldung
```

```
            result = getMessageText (1002, ...+"|"+...+"|"+...);
        }
        catch (CbaException se)
        {
            // Durchreichen einer bereits behandelten Exception
            throw se;
        }
        catch (Exception e)
        {
            // Behandlung einer neuen Exception
            setMessage(1000, e.Message, "E", "eâ€□");
        }
        finally
        {
            // ... abschliessende Massnahmen
        }
        return result;
    }
```