Regelausführung selektiv ausschalten

Definition

Möchte man verhindern, dass bestimmte Regeln oder Teile davon ausgeführt werden, kann man dies mithilfe regelübergreifender (ThreadLocal) Variablen erreichen.

Beispiel

Ein typischer Anwendungsfall dafür ist die gezielte Aktualisierung einzelner Felder in Businessobjekten aus einer Regel hinaus, bei der man z.B. aus Performancegründen oder um Endlosschleifen durch sich gegenseitig aktualisierende Regeln bzw. Businessobjekte zu vermeiden, verhindern will, dass die Speichernregeln (UpdateRule, UpdateFinalRule) des zu aktualisierenden Businessobjektes ausgeführt werden. Natürlich ist bei Einsatz der im Folgenden beschriebenen Methode darauf zu achten, dass man die fachliche Logik der Regeln nicht aushebelt, es ist also mit Bedacht vorzugehen.

Einführung einer ThreadLocal-Variable in der zu deaktivierenden Regel

Zu deaktivierende Regel

```
public class MyUpdateRule implements UpdateRule, StateChangeFinalRule {

        //definition of thread local variable
    public static final ThreadLocal<Boolean> ACTIVE = new ThreadLocal<Boolean>() {
        protected Boolean initialValue() {return Boolean.TRUE;};
    };

    public void update(UpdateContext context) throws BusinessException {
                //check if rule is active or not (in the running thread)
        if (ACTIVE.get()) {
            MyBusinessObject mbo = context.getBusinessObject(MyBusinessObject.class);
                        ...
        }
    }
}
```

ThreadLocal-Variablen sind lokal auf den Thread bezogen, dass heisst die Deaktivierung einer Regel bezieht sich immer nur auf den aktuellen Thread bzw. Benutzerkontext. Die Regel wird also nicht generell deaktiviert, wie z.B. über das Aktiv-Flag der Regel.

Deaktivierung aus aktualisierender (anderer) Regel heraus

Deaktivierende Regel

```
 public class SomeOtherRule {

    ...
                ...
                MyBusinessObject mbo = QueryProvider.getById(MyBusinessObject.class, id);
                mbo.setMyField(value);
                 try {
                        //deactivate rule
            MyUpdateRule.ACTIVE.set(Boolean.FALSE);
                        //content of if-block of update rule (see above) will not be executed this time
            BusinessObjectProvider.update(mbo);
                 } finally {
                        //activate rule again
                        MyUpdateRule.ACTIVE.set(Boolean.TRUE);
                }
                ...
        ...
}
```

Es ist darauf zu achten, dass die Deaktivierung zuverlässig wieder zurückgesetzt wird (auch im Fehlerfall), daher sollte dies immer wie in diesem Beispiel im finally-Block geschehen. Die Aktivierung und Deaktivierung befindet sich in diesem Beispiel also in einem try-catch-finally-Block, um sicherzustellen, dass selbst nach Auftreten einer Exception in BusinessObjectProvider.update() die Regel anschliessend wieder aktiviert wird.