Lazy References clearen

1.) Erst mal ganz simpel (und manuell):

Die "Lazy" Klasse hat eine Methode #clear.
Wenn man die aufruft, ist die in der Lazy Reference gehaltene Referenz entfernt und nur noch die ID wird gehalten, damit die Instanz bei Bedarf wieder geladen werden kann.

Wichtiges Hintergrundwissen:
So ein clear heißt aber nicht, dass die referenzierte Instanz gleich aus dem Speicher verschwindet. Das macht ja nur der Garbage Collector der JVM.
Die Referenz ist sogar noch an einer weiteren Stelle registriert, nämlich in einem globalen Verzeichnis ("SwizzleRegistry"), in dem jede MicroStream bekannte Instanz mit ihrer ObjectId eineindeutig registriert ist ("eineindeutig" heißt: OID -> Instanz und auch Instanz -> OID).
Das heißt: Wenn man so eine Referenz cleart, aber kurz darauf wird die Lazy Reference wieder abgefragt, muss wahrscheinlich gar nichts aus der Datenbank Datei geladen werden, sondern es wird einfach nur wieder die Referenz aus der SwizzleRegistry wiederhergestellt.
Trotzdem ist die SwizzleRegistry kein Memory-Leak, denn sie referenziert die Instanzen nur via "WeakReference". Bei Bedarf gern googeln. Kurz gesagt: Wenn eine Instanz nur noch "weak" referenziert wird, räumt der JVM GC sie trotzdem auf.

2.) Manuell ist doof:

Damit die Lazy References nicht manuell gemanaget werden müssen, sondern das ganze automatisch geht, gibt es folgenden Mechanismus:
Jede "Lazy" Instanz hat einen "lastTouched" Timestamp.
Bei jedem .get() Aufruf wird der auf die aktuelle Zeit gesetzt.
Damit kann man feststellen, wie lang eine LazyReference schon nicht mehr verwendet worden ist, d.h. ob sie überhaupt noch gebraucht wird.

Der, der das prüft, ist der "LazyReferenceManager".
Standardmäßig ist der nicht aktiviert, weil der einen eigenen Thread braucht und weil ich es problematisch finde, wenn Frameworks ungefragt Threads starten.
Der ist aber schnell aktiviert:

Etwa so:

```
// Beispiel für Timeout von Lazy References nach 10 Minuten ohne Zugriff auf die Referenz.
LazyReferenceManager.set(LazyReferenceManager.New(10*60*1000).start());
```

10 mal 60 mal 1000 Millisekunden als Timeout heißt: Nach 10 Minuten ohne Benutzung wird eine Lazy Referenz gecleart.

Was man hier einstellt ist geschmackssache und hängt stark von der Art der Anwendung ab.
Für manche mögen 10 Sekunden schon eine riesige Zeitspanne sein, bei der der Memory zu schnell vollläuft, weil ständig so viel geladen, einmal verarbeitet und dann ignoriert wird.
Für andere sind vielleicht 10 Minuten noch zu kurz, weil riesige Datenmengen geladen werden, die dann möglichst mehrere Stunden oder Tage im Speicher vorgehalten werden sollen.
Knifflig.

Wir könnten natürlich noch eine hübsch bequeme Methode

```
LazyReferenceManager.setDefault();
```

machen. Aber was nehmen wir als Default Timeout?
Für den Moment sollte es zumutbar sein, irgendein "X*60*1000" hinzuschreiben und los gehts.

Weitere Überlegungen stehen Z.B. in https://www.xdevissues.com/browse/MS-131