# 1 Content



# 2 Ojb.Net

| | |
|---|---|
| Websites | http://ojb-net.sourceforge.net |
| Allgemeines | Ojb.Net ist die Portierung des Apache Projektes ojb. |
| Lizenz | LGPL: Kann für Open Source und Kommerzielle Software gratis verwendet werden.<br>Der Source Code ist vorhanden |
| Bekanntheit / Verbreitung | Die aktuelle Version (0.7.1410) befindet sich noch im Beta Stadium.<br>ojb.net scheint nicht sonderlich verbreitet zu sein. Google liefert nur 894 Treffer.<br>Der Release-Intervall ist ungefär ein Monat. |
| Dokumentation | Die Dokumentation ist sehr spärlich und besteht nur aus einem Tutorial, welches nur die wichtigsten Funktionen von Ojb.Net erklärt.<br>Ein Forum ist vorhanden, wird jedoch nur selten genutzt (Letzer Eintrag ist 2 Wochen vergangen). |
| Unterstützte Datenbanken | • MS SQL Server 2000 |
| Kompatibel zu Struktur und Ablauf von pmMDA | Ja, ausser:<br>Zusammengesetzte Primärschlüssel werden von der Standartimplementation nicht unterstützt. Eine Eigene der Basisklasse für DataObjects würde vermutlich Abhilfe schaffen.<br>Bei der Vererbungshierarchie gelten dieselben Bemerkungen wie bei NHibernate. |
| Konfigurierbarkeit des Mappings | Das Mapping wird in einer xml-Datei spezifiziert. Die Datenbanktypen werden in diesem Repository angegeben.<br>Die Semantik des Repository stimmt weitgehend mit dem ojb-repository überein; nur der Syntax ist unterschiedlich. |
| Transaktionsmanagement | Das Transaktionsmanagement wird von Ojb.Net unterstützt.<br>Allerdings ist es nicht möglich eigene Transaktionsobjekte zu erstellen. Stattdessen können selbst geschriebene Methoden mit Hilfe von Attributen als Transaktion ausgeführt werden. |
| Queries | Queries wurde von Ojb.Net über Criterias realisiert. Criterias werden unabhängig von dem Datenbankprovider erstellt und von Ojb in SQL Code umgesetzt.<br>OQL wird nicht unterstützt. |
| Cache | Eine Interface-Definition für Cache-Funktionen und eine Standartimplementation ist vorhanden. |
| Code features | Dataobjekte müssen in Ojb.Net immer von der Klasse MarshalByRefObject abgeleitet sein. Somit können Dataobjekte als Basisklasse nur andere Dataobjekt-Klassen besitzen.<br>Anderenfalls müsste das Domain-Model über Interfaces nachgebildet werden, was zu einer grossen Divergenz mit der pmMDA-java Implementation ergeben |

| | |
|---|---|
| | würde.<br>Assoziationen: Es werden alle möglichen Arten unterstützt (1...1; 1...n; n...m) |

# 3 NHibernate

| | |
|---|---|
| Websites | http://nhibernate.sourceforge.net |
| Allgemeines | NHibernate ist die Protierung des Java Projektes Hibernate.<br>Hibernate is a powerful, ultra-high performance object/relational persistence and query service for Java. Hibernate lets you develop persistent classes following common Java idiom - including association, inheritance, polymorphism, composition and the Java collections framework. The *Hibernate Query Language*, designed as a "minimal" object-oriented extension to SQL, provides an elegant bridge between the object and relational worlds. Hibernate also allows you to express queries using native SQL or Java-based Criteria and Example queries. |
| Lizenz | LGPL: Kann für Open Source und Kommerzielle Software gratis verwendet werden.<br>Der Source Code ist vorhanden |
| Bekanntheit / Verbreitung | Die aktuelle Version (0.6.0.0) befindet sich noch im Beta Stadium.<br>Die Suche auf Google nach NHibernate liefert 23'500 Treffer. Somit scheint NHibernate weit verbreitet zu sein.<br>Das Forum scheint häufig besucht zu werden (Alleine gestern gab es min. 8 Einträge) |
| Dokumentation | Tutorial, NHibernate Reference (noch in Bearbeitung) |
| Unterstützte Datenbanken | • DB2<br>• Firebird<br>• MS SQL 2000<br>• MS SQL 7<br>• Oracle<br>• Oracle9<br>• MySQL<br>• PostgreSQL<br>• Sybase<br>Weitere Dialekte können durch Implementation einer abstrakten Basisklasse realisiert werden. |
| Kompatibel zu Struktur und Ablauf von pmMDA | Falls NHibernate als OR Mapper verwendet werden würde müsste das MetaModell von pmMDA angepasst werden: Die MetaClass Klasse muss dann nicht nur seine Basisklasse und –Interfaces kennen (ancestors), sondern auch die Klassen welche von der aktuellen Klasse ableiten (derived classes). Diese Informationen werden für die Konfigurations-Dateien benötigt.<br>Diese Erweiterung aber auch für ojb (Java) gemacht werden, falls dort das Tag extent-class verwendet wird. Allerdings hat pmMDA Vererbungen noch nicht implementiert, so dass dies nicht sicher ist.<br>Ein Problem stellen 1..n Assoziationen dar. In pmMDA kann die Collection Klasse bzw. das Interface, über welches auf die Properties zugegriffen werden kann, konfiguriert werden. NHibernate benutzt jedoch eigene Collection Klassen (Map, Set, List, …), welche nicht konfiguriert werden können.<br>Zusammengesetzte Primärschlüssel werden unterstützt. |
| Konfigurierbarkeit des Mappings | Das Mapping ist sehr flexibel. So wird standartmässig pro DataObject Klasse eine Mapping Datei (Klassenname.hmb.xml) erstellt und als eingebettete Ressource in das Assembly kompiliert. Das Mapping könnte jedoch auch in einer (1) externen XML-Datei gemacht werden.<br>Das Format unterscheidet sich sehr stark vom ojb-Repository. Jedoch sollten die gleichen Funktionen realisierbar sein.<br>Das Mapping kann sehr flexibel aufgebaut werden. So können one-to-many und |

| | |
|---|---|
| | many-to-one Assoziationen invers oder in einer eigenen Tabelle gespeichert werden. |
| Transaktionsmanagement | Transaktionen werden unterstützt. Sie sind, zumindest auf den ersten Blick, sehr ähnlich wie die ojb (ohne .NET!) Transaktionen. |
| Queries | Hibernate benutzt eine eigene Hibernate Query Language (HQL).<br>Zudem werden Criterias unterstützt (Objektbasierte API), welche jedoch noch nicht so mächtig ist, wie die HQL. |
| Cache | Definitionen und Standartimplementationen sind vorhanden. |
| Code features | Vererbungen werden unterstützt. Wahlweise kann zwischen einer Tabelle für alle abgeleiteten Klassen und einer Tabelle pro Klasse gewählt werden.<br>Alle abgeleiteten klassen können im der selben XML-Datei spezifiziert werden.<br>Es werden alle möglichen Arten von Assoziationen unterstützt (1...1; 1...n; n...m). Diese können auch sehr unterschiedlich (=flexibel) gespeichert werden. |
| Indexed Properties | NHibernate implementiert eigene Collections für 1..n, n..1 und n…m Assoziationen. Daraus ergibt sich, dass die Data Objekte für ihre Indexed Properties ein spezielles Property brauchen, mit welchem NHibernate die Items des Indexed Property als Collection setzen kann.<br>Ein Beispiel dazu ist im Beispielprojekt unter Planet.Satellites zu finden. |
| one-to-one | 1..1 Properties zu anderen Business Objekten können nur realisiert werden, wenn beide an der Assoziation beteiligen Data Objekte die gleiche Id besitzen. Dies führt zu Problemen wenn die beiden Data Objekte dieselbe Basisklasse haben (Dann darf die Id nicht gleich sein!). In diesem Fall muss die one-to-one Assoziation durch eine many-to-one Assoziation ersetzt werden. |

# 4 Gentle.Net

| | |
|---|---|
| Websites | http://www.mertner.com/confluence/display/Gentle/Home |
| Allgemeines | Gentle.NET is an RDBMS independent object persistence framework. It features automatic SQL generation and object construction, an SQL factory for creating custom queries, DataView construction helpers, excellent performance and reasonably complete docs. |
| Lizenz | LGPL: Kann für Open Source und Kommerzielle Software gratis verwendet werden.<br>Der Source Code ist vorhanden |
| Bekanntheit / Verbreitung | Die aktuelle Version ist 1.2.0 und in Sourceforge als Production/Stable eingestuft. Zu dem Projekt gibt es ein Forum, das ziemlich aktiv ist. Ausserdem führt die Community eine eigene Website (http://www.mertner.com/confluence/display/Community/Home).<br>Eine Suche auf Google nach „Gentle.NET" ergab 5630 Treffer. |
| Dokumentation | • Ausführliche API Dokumentation<br>• User Guide auf der Homepage<br>• Forum<br>• Community Website<br>Ausserhalb der API Dokumentation sind nur wenig nützliche Informationen zu finden. → schwache Tutorials, Beispiele und Einführungen |
| Unterstützte Datenbanken | • Firebird<br>• MySQL<br>• PostgreSQL<br>• Oracle<br>• MS Access, SQL Server |

| | |
|---|---|
| | • Sybase |
| Kompatibel zu Struktur und Ablauf von pmMDA | Generell: Ja<br>Vererbung: Nein<br>Collections für 1..n und n..m Assoziationen sind vorgegeben. Dies führt zum selben Problem wie bei NHibernate.<br>Weitere Abklärungen nötig… |
| Konfigurierbarkeit des Mappings | Das Mapping wird über Attribute im Source Code spezifiziert. Die Datenbanktypen werden automatisch gemappt.<br>→ Es ist keine Konfigurationsdatei für das Mapping nötig. In der Konfigurationsdatei stehen nur der Connection String zu der Datenbank. |
| Transaktionsmanagement | Das Transaktionsmanagement wird von Gentle.Net unterstützt.<br>Dazu wird die Transaction Klasse zur Verfügung gestellt. |
| Queries | In Gentle.NET existieren die Klassen SqlBuilder und SqlStatement um Queries zu generieren.<br>OQL wird nicht unterstützt. |
| Cache | Eine Default Caching Strategie lässt sich über die Gentle.NET Konfiguration festlegen. Klassenspezifisch lässt sich die Caching Strategie über Attribute steuern. |
| Code features | Die Klassen müssen nicht von einer Klasse abgeleitet sein. Für einige Funktionen wird verlangt, dass das Interface IPersistent implementiert wird.<br>Zur Vereinfachung existiert die Klasse Persistent, welche eine Default-Implementation des IPersistent Interface enthält.<br>Assoziationen werden alle möglichen Arten unterstützt (1...1; 1...n; n...m). Für n…m Assoziationen muss die GentleList verwendet werden, für die restlichen Assoziationen kann auch eine IList verwendet werden. |
| Vererbung | Gentle.NET unterstützt im Moment nur das „one-class-one-table"-Verfahren. Das beseutet: Ist eine Klasse B von der Klasse A abgeleitet, werden die Tabellen TA und TB erwartet. Dabei enthält die Tabelle TB auch die Spalten von TA. Ein Objekt der Klasse B wird nur in TB gespeichert. Somit liefert eine Anfrage für alle Objekte der Klasse A nicht die Objekte der Klasse B. |

# 5 Fazit

Alle OR Mapper sind sehr komplex. Eine Einarbeitung benötigt viel Zeit; mehr als wir uns vorgestellt hatten.

Alle von uns angeschauten OR Mapper funktionieren. Nur bei ojb.net mussten einige kleine Anpassungen am Source vorgenommen werden (zum Beispiel um Tabellenname wie +a.b.c.d" zu unterstützen).

Bei Ojb.net ist die Dokumentation sehr spärlich ist und es existiert nur eine kleine Communitiy. Zudem scheint das Projekt nicht sehr aktiv zu sein. Zudem musste der Source Code an einigen Stellen angepasst werden um pmMDA kompatibel zu sein. Ojb.net gibt zudem die Basisklasse aller Data Objekte vor (MarshalByRefObject).

Deshalb ist eine Benutzung von Ojb.net riskant und wir raten davon ab.

NHibernate ist sehr flexibel und das bei weitem grösste OR Mapper Projekt (85'000 Source Zeilen). Dadurch ist die Handhabung komplex. Die Dokumentation ist sehr ausführlich. Dies kommt daher, dass NHibernate eine Portierung des Java Projektes Hibernate ist. Da die Formate der Konfigurationsdateien 1:1 übernommen wurden existieren viele Beispiele.

Die Konfiguration über XML-Dateien ist sehr flexibel. So können diese in das Assembly kompiliert, oder extern eingebunden werden.

Obwohl NHibernate noch im Beta Stadium ist, wurden keine Fehler entdeckt.

Gentle.NET ist angenehm einfach zu konfigurieren, da alles über Code-Attribute gesteuert wird, und kein eigenes XML-Format definiert wurde. Durch diese Attributierung wird Gentle.NET etwas unflexibel. Vererbungen werden im Moment nur im „one class – one table" Verfahren unterstützt. Sehr angenehm ist das automatische Umwandeln von Datenbank-Typen in .NET Datentypen.

Es ist der einzige OR Mapper der den Status Stabil erreicht hat.

Weder bei NHibernate noch bei Gentle.NET müssen die Data Objekte ein Interface implementieren oder von einer Klasse ableiten. Dafür werden bei diesen beiden OR Mapper die Collection Klassen und Interfaces vorgegeben. Normalerweise könnten diese Typen durch pmMDA konfiguriert werden.

Da bei Gentle.NET Vererbungen in verschiedene Tabellen nicht implementiert hat und der Data Objekt Code mit Attributen versehen werden muss, tendieren wir im Moment zu NHibernate.

Damit pmMDA in java und .NET korrelieren wäre eine mögliche Variante ojb im pmMDA mit Hibernate, bzw. NHibernate zu ersetzen.

# 6 Weiteres Vorgehen

Die bestehenden (.NET) Cartridges in pmMDA werden auf NHibernate angepasst. Danach wird ein Client geschrieben, der CRUD (create, read, upddate, delete) auf den automatisch erstellten Data Objekte zu Testzwecken durchführt.

Danach wird das DOG in .NET nachgebildet. DOG soll (zudem?) als eine Facade für den Zugriff auf die OR-Mapper Funktionen fungieren. Dadurch könnte zu einem späteren Zeitpunkt der OR Mapper ausgetauscht werden (z.B: durch Object Spaces).