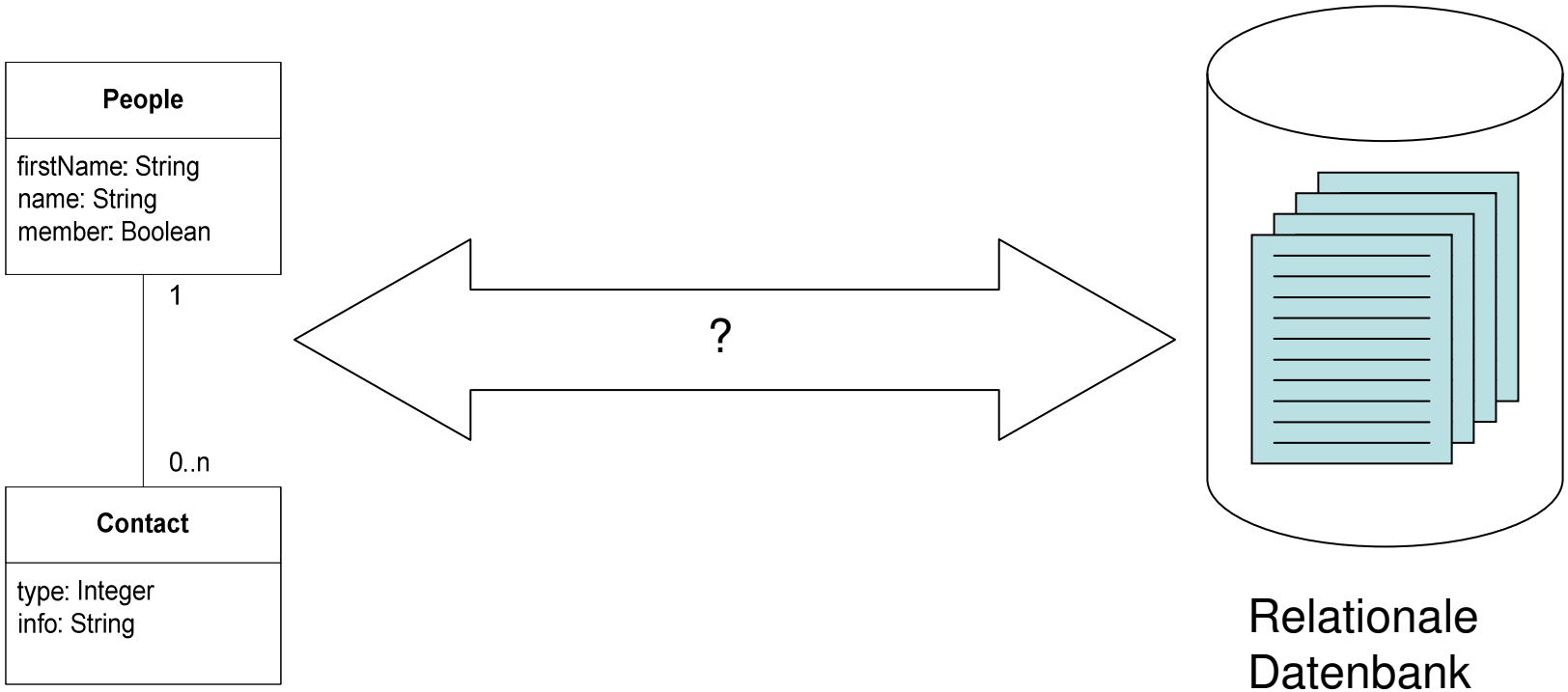
O/R Mapping

Henning Schmiedehausen
hps@intermeta.de
KNF Kongreß 2006, Nürnberg
26. Oktober 2006

Was ist das?

Objektorientierte Entwicklung



Wozu brauche ich das?

- “Objekt – Relationales Mapping”
- “Impedanzproblem”
- Resourcenverwaltung
- Kapselung der Datenhaltung

O/R Mapping kurzgefaßt

“Object-Relational Mapping (O/RM), ist eine Programmiertechnik, die Daten-banken mit objektorientierten Programmierkonzepten verbindet und dadurch eine ‘virtuelle Objektdatenbank’ erzeugt” – Wikipedia

Wer hat’s erfunden?

- Chen / Yourdon – Entity model
- Fowler / Evans – Domain model
- Scott Ambler – Agile Database Development
- Apple – Enterprise Objects Framework

Beispiel: Datenbank Zugriff

```
public String getPersonInformation(final String personName) {
  Connection c =
    DriverManager.getConnection("jdbc:postgresql://localhost/persondb",
                                "user", "pass");
  Statement s = c.createStatement();
  ResultSet rs = s.executeQuery(
           "SELECT * from PERSON_INFORMATION where person_name = '"
           + personName + "'");
  if (rs.next()) {
    String personInformation = rs.getString(4);
    return personInformation;
  }

  return null;
}
```

Datenbankspezifisch?

```
public String getPersonInformation(final String personName) {
  Connection c =
    DriverManager.getConnection("jdbc:postgresql://localhost/persondb",
                                "user", "pass");
  Statement s = c.createStatement();
  ResultSet rs = s.executeQuery(
           "SELECT * from PERSON_INFORMATION where person_name = '"
           + personName + "'");
  if (rs.next()) {
    String personInformation = rs.getString(4);
    return personInformation;
  }

  return null;
}
```

Tabellenstruktur?

```
public String getPersonInformation(final String personName) {
  Connection c =
    DriverManager.getConnection("jdbc:postgresql://localhost/persondb",
                                "user", "pass");
  Statement s = c.createStatement();
  ResultSet rs = s.executeQuery(
            "SELECT * from PERSON_INFORMATION where person_name = '"
            + personName + "'");
  if (rs.next()) {
    String personInformation = rs.getString(4);
    return personInformation;
  }

  return null;
}
```

Fehler und Resourcen?

```
public String getPersonInformation(final String personName) {
  Connection c =
    DriverManager.getConnection("jdbc:postgresql://localhost/persondb",
                                "user", "pass");
  Statement s = c.createStatement();
  ResultSet rs = s.executeQuery(
           "SELECT * from PERSON_INFORMATION where person_name = '"
           + personName + "'");
  if (rs.next()) {
    String personInformation = rs.getString(4);
    return personInformation;
  }

  return null;
}
```

Sicherheit?

```
public String getPersonInformation(final String personName) {
  Connection c =
    DriverManager.getConnection("jdbc:postgresql://localhost/persondb",
                                "user", "pass");
  Statement s = c.createStatement();
  ResultSet rs = s.executeQuery(
           "SELECT * from PERSON_INFORMATION where person_name = '"
           + personName + "'");
  if (rs.next()) {
    String personInformation = rs.getString(4);
    return personInformation;
  }

  return null;
}
```

Generische Typen?

```
public String getPersonInformation(final String personName) {
  Connection c =
    DriverManager.getConnection("jdbc:postgresql://localhost/persondb",
                                "user", "pass");
  Statement s = c.createStatement();
  ResultSet rs = s.executeQuery(
           "SELECT * from PERSON_INFORMATION where person_name = '"
           + personName + "'");
  if (rs.next()) {
    String personInformation = rs.getString(4);
    return personInformation;
  }

  return null;
}
```

Warum sollte mich das jucken?

- “so haben wir schon immer programmiert”
- “wird doch nur intern benutzt”
- “Rechner sind schnell genug”
- “ist doch nur ein Mini-Programm für mich”
- “funktioniert doch”
- “Zeitdruck”

Risiken

- Sicherheit (CGI / Netzwerkcode)
- Schlechte Portierbarkeit
- Schlechte Wartbarkeit
- Schlechte Lesbarkeit
- Schwer zu findende Resource-Löcher

➔ Wartung frißt die “gefühlten Vorteile” auf

O/R Mappers

- Gibt es in verschiedenen “Härtegraden” für die meisten Programmiersprachen
  - .NET: Genome, .NET Data Objects, OpenAccess
  - Perl: Class-DBI
  - PHP: Propel, EZPDO, ActiveRecord
  - Objective-C: Enterprise Object Framework
  - Ruby: ActiveRecord, Og, Rubernate

O/R Mappers für Java

- Java
  - Castor
  - **Apache Cayenne**
  - CrossDB
  - Enterprise Objects Framework (WebObjects)
  - FireStorm DAO
  - **Hibernate (RedHat / JBoss)**
  - Hydrate
  - **Apache iBATIS**
  - intelliBO
  - Java Data Objects (JDO 1 + JDO 2)
  - JDX
  - JPA
  - JPOX (JDO 2 Implementation)
  - Lychee
  - OpenAccess
  - Apache DB OJB
  - SimpleORM
  - Speedo
  - Oracle Toplink + TopLink Essentials
  - Apache DB Torque
  - JDBCPersistence
  - **Mehr…**

Ab hier: Java

O/R Mapping in der Java-Welt

- O/R Mapping wird von Hibernate dominiert
- EJB 3 and JDO 2
- JPA wird der allgemeingültige Standard werden
- Oracle TopLink und Apple Enterprise Objects
  - (TopLink Essentials ist Open Source Referenz-implementation (RI) für JSR-220 (EJB 3.0 Spezifikation))

- Warum noch was anderes anschauen?

Warum nicht?

- EJB 3 und JDO 2 sind kompliziert
  - EJB3 spec: 818 Seiten! (Basis: 59)
  - JDO2 spec: 332 Seiten
- Hibernate ist unter LGPL
  - Rechtliche Probleme beim Binden mit eigenem Code
- JDO ist ‘politisch’
- EJB3 ist relativ neu

Also zurück zum Selberrollen?
Ich will doch nur Daten für
meine (Web)Applikation!

Daten vs. Objekte

- O/R Mapper
  - Hibernate (JBoss / RedHat)
  - Apache Cayenne
- SQL Maps/Data Mapper
  - Apache iBATIS

Gemeinsamkeiten

- Datenbanken: MySQL, PostgreSQL, Oracle, HSQLDB, Apache Derby
- Mapping Definitionen in XML Dateien
- 1:1, 1:n, m:n Relationen
- Primärschlüsselerzeugung durch Datenbank oder Applikation

- “Half day rule”

Hibernate

- Version 3.2.1 (16.11.06)
- Transparente Persistenz
- HQL und Criteria Queries
- J2EE Integration
- OO-Features: Polymorphismus, Vererbung, Zusammenstellungen
- JDK 1.5 Annotations

Hibernate

- Pros:
  - Industriestandard (Bücher, Tools usw.)
  - JPA, EJB3, J2EE Features
  - POJO Support
- Contra:
  - Steile Lernkurve
  - Lizenz (LGPL)
  - Resourcebedarf

Apache Cayenne

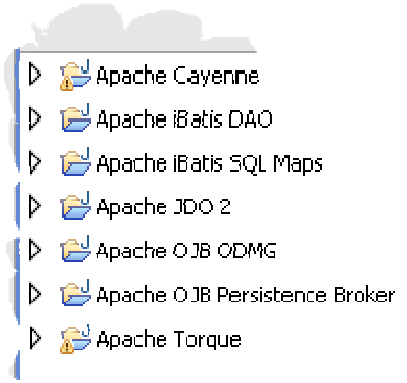
- Apache Incubator Podling
- Version 2.0.1 (7.10.06)
- GUI Tool für Mappings enthalten
- Unterstützt Apache ant, Apache Maven
- Konzepte ähnlich WebObjects EOF
- J2EE: JNDI, Managed Transactions

Apache Cayenne

- Pros:
  - GUI modeler enthalten
  - Gute Dokumentation
  - Schlanke Architektur
- Cons:
  - Objecten müssen von `DataObject` ableiten
  - Keine JTA Unterstützung
  - Keine Standard APIs

Beispielcode

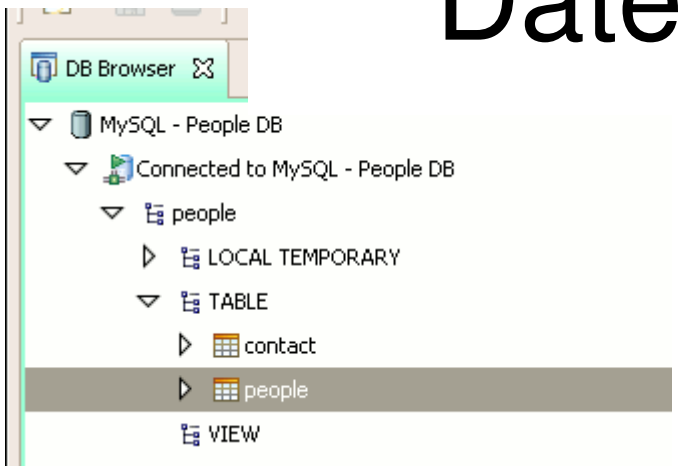
- Vortrag kratzt nur an der Oberfläche
- Beispielcode kann von meiner Homepage geladen werden:

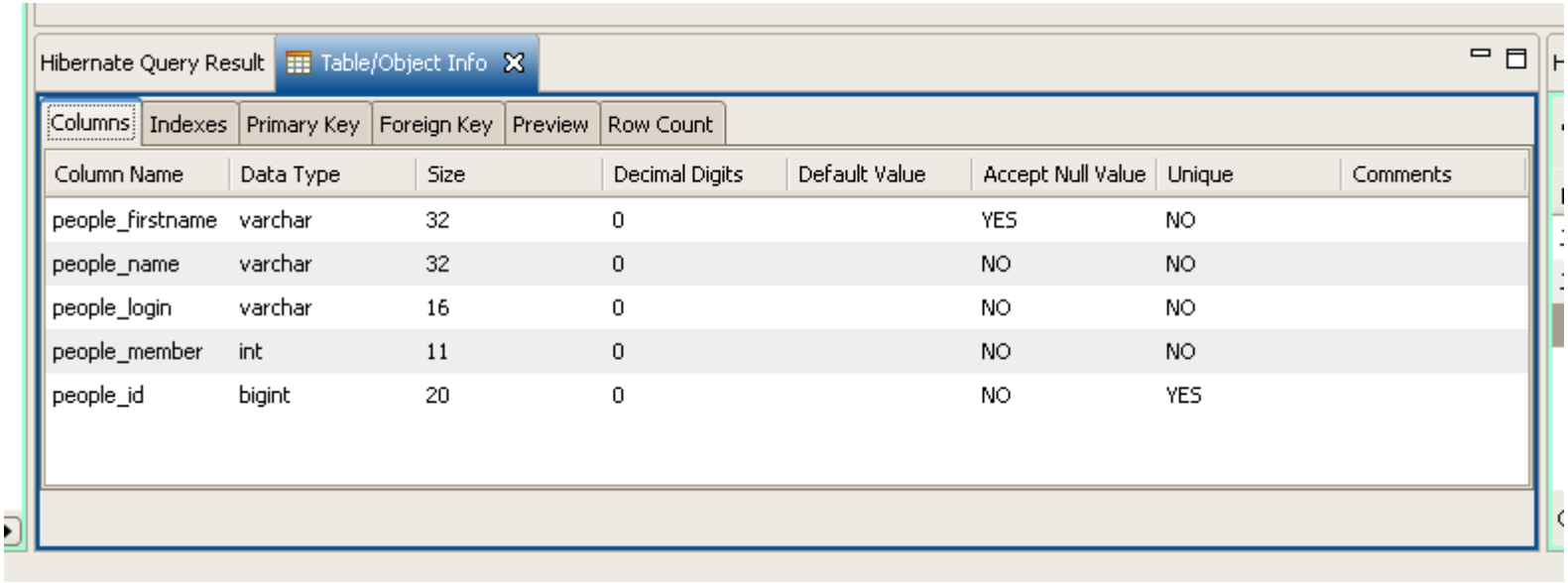


http://henning.schmiedehausen.org/or-mappers/

http://svn.softwareforge.de/svn/opensource/talks/or-mappers/

Datenbank Tabelle





Column Name	Data Type	Size	Decimal Digits	Default Value	Accept Null Value	Unique	Comments
people_firstname	varchar	32	0		YES	NO	
people_name	varchar	32	0		NO	NO	
people_login	varchar	16	0		NO	NO	
people_member	int	11	0		NO	NO	
people_id	bigint	20	0		NO	YES	

Java Objekt

```
public class People {
  private long id;
  private String name;
[…]
  public long getId() { return this.id; }

  public void setId(long id)
      {  this.id = id; }
[…]
  public String getName() { return this.name; }

  public void setName(String name)
      { this.name = name; }
}
```

Primärschlüssel Suche

- Hibernate:

```
public People retrieve() {
  People user = (People)
    session.get(People.class, 2L);
  return user;
}
```

- Cayenne:

```
public People retrieve() {
  People user = (People) DataObjectUtils
   .objectForPK(dataContext, People.class, 2);
  return user;
}
```

Objekte ändern

- Hibernate:

```
Transaction tx = session.beginTransaction();
People people = retrieve();
people.setName("Meier");
session.flush();
tx.commit();
```

- Cayenne:

```
People people = retrieve();
people.setName("Meier");
dataContext.commitChanges();
```

And Now…
…for Something Completely Different

Apache iBATIS

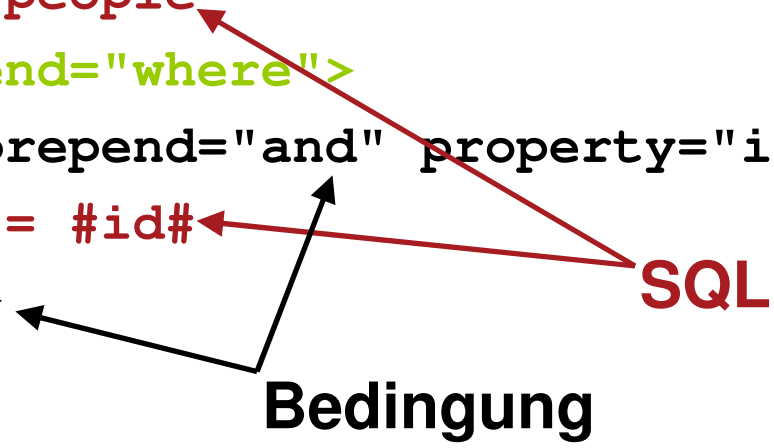
- Aktuelle Version is 2.2.x
- Daten Mapper, kein Objekt Mapper
- Mappt SQL Abfragen
- J2EE Features: JNDI, JTA
- optionale DAO (data access objects) Schicht (unterstützt iBatis SQLMAP, Hibernate, JDBC)

Was ist der Vorteil?

- O/R Mapper: Objektbeziehungen
- iBATIS: Datenabfragen
- Datenbanken müssen nicht 3NF haben
- Vorhandene Datenbestände
- Portierung von Applikationen (PHP, perl)

Eine iBATIS Abfrage

```
<select id="peopleQuery" parameterClass="people"
                         resultMap="peopleResult">
  select * from people
  <dynamic prepend="where">
    <isNotNull prepend="and" property="id">
      people_id = #id#
    </isNotNull>
  </dynamic>
</select>
```



Data Mapping

- Eingabe Parameter:

```
<typeAlias alias="people“
           type=“ibatis.People"/>
```

- Ausgabe Parameter:

```
<parameterMap id=“peopleResult“
           class=“people">
    <parameter property=“name“
           jdbcType="VARCHAR" />
    <parameter property="id"/>
</parameterMap>
```

Datenabfrage

- Ein Objekt mit Primärschlüssel

```
People select = new People();
select.setId(2L);
People user = (People)
  sqlMap.queryForObject("peopleQuery", select);
```

- Alle Objekte abfragen

```
List<People> select = (List<People>)
  sqlMap.queryForList("peopleQuery", null);
```

Welches Projekt verwenden?

- Diese Tabelle ist stark subjektiv! YMMV!

Was	Wann
Hibernate, Cayenne	…wenn man besser Java als SQL programmiert
iBATIS	…wenn man besser SQL als Java programmiert
Hibernate	Wenn Standards eine Rolle spielen
Cayenne, iBATIS	Wenn Lizenzfragen eine Rolle spielen

Anlaufstellen

- Folien (KNF, ApacheCon), Beispielcode
  - http://henning.schmiedehausen.org/or-mappers/
  - http://svn.softwareforge.de/svn/opensource/talks/or-mappers/
- O/R Mappers Homepages:
  - http://www.hibernate.org/
  - http://incubator.apache.org/cayenne/
  - http://ibatis.apache.org/
- Apache DB Projekt
  - http://db.apache.org/

Fragen?

Vielen Dank für die Aufmerksamkeit!