

# Agenda

## Einstieg

### ORM

JPA

Hibernate

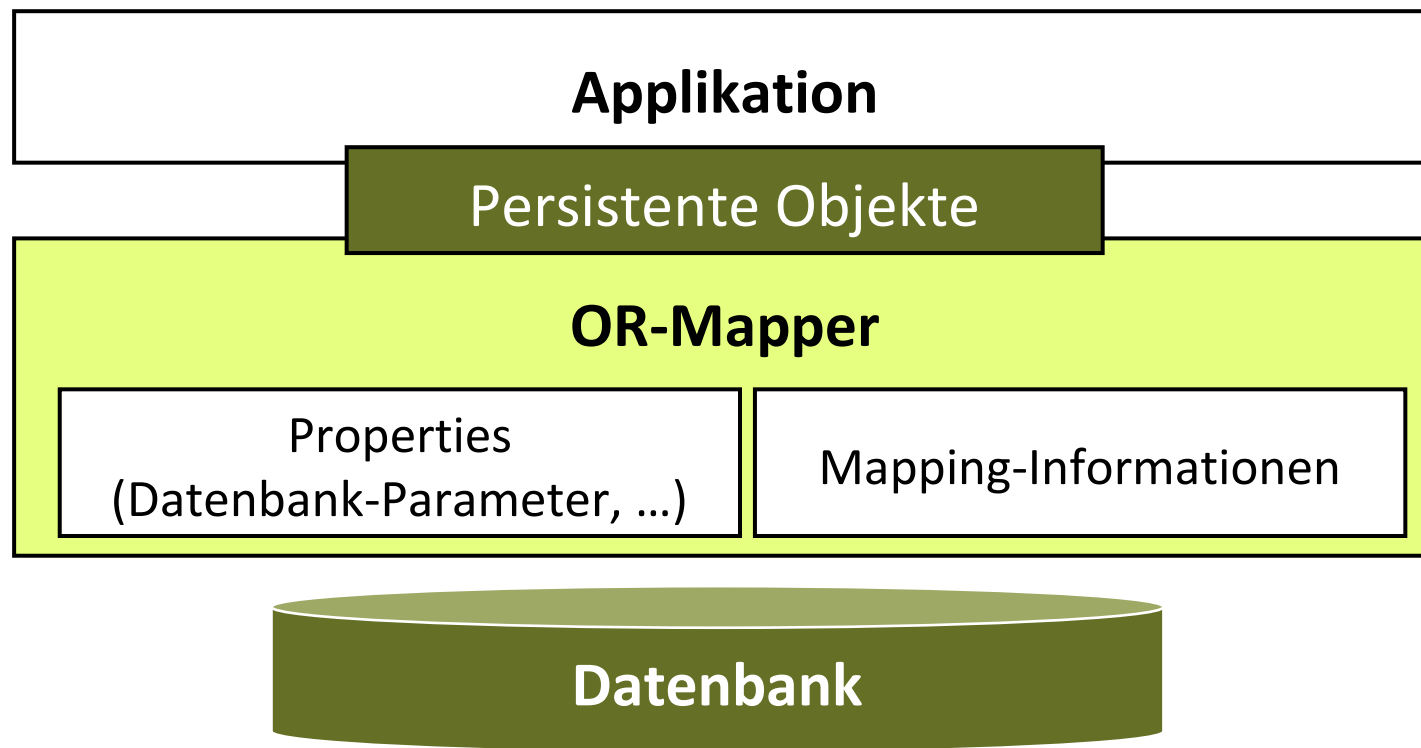
Andere Provider

# Transient und Persistent

- Ein normales Objekt, wie es in der JVM vorkommt, nennt man **transient**.
  - ◆ Transiente Objekt leben nur in der JVM und sterben, wenn die JVM (unerwartet) beendet wird oder das Objekt nicht mehr referenziert wird und vom GC weggeräumt werden kann.
- **Persistente Objekte** haben eine Abbildung in der Datenbank.
  - ◆ Ein Schlüssel identifiziert sie eindeutig.
  - ◆ Sie sind transaktional.

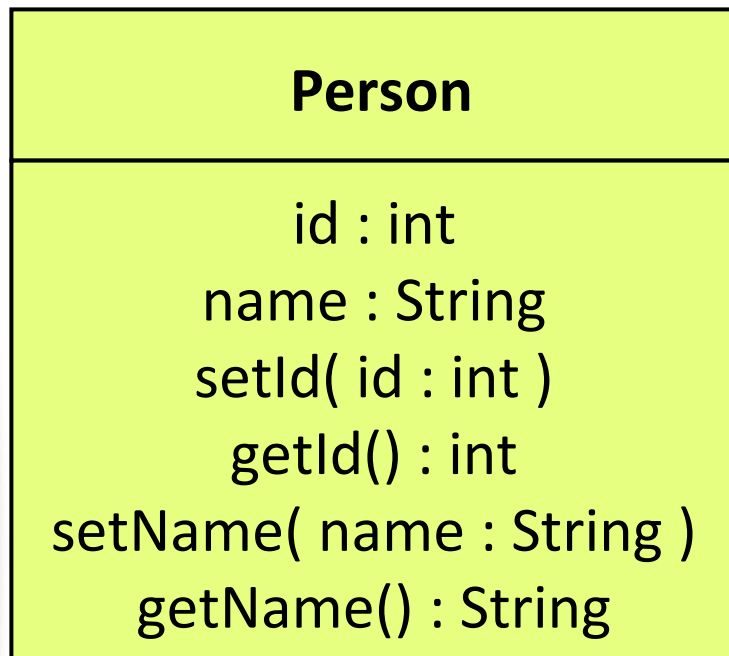
# Arbeitsweise eines O/R-Mappers

- **O/R** steht für **Objekt-Relationales** Mapping, also die Abbildung der Java-Objekte auf Datenbank-Relationen.




# Abbildungen der Objekte auf Relationen

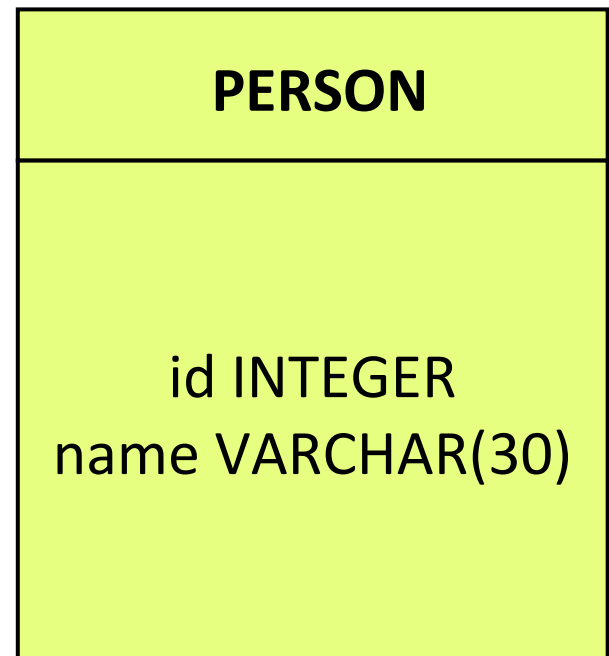
In der JVM



Hier  
arbeitet  
der  
OR-Mapper



In der Datenbank



# Wie Java Objekte persistieren?

- **JDBC:**

Persisting the Object Graph of Book Object

```
public class BookStoreService {
    private Connection connection = null;

    public void persistObjectGraph(Book book) {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/bookstore", "root", "password");

            PreparedStatement stmt = connection.prepareStatement("INSERT INTO PUBLISHER (CODE, PUBLISHER_NAME) VALUES (?, ?)");
            stmt.setString(1, book.getPublisher().getCode());
            stmt.setString(2, book.getPublisher().getName());
            stmt.executeUpdate();

            stmt.close();

            stmt = connection.prepareStatement("INSERT INTO BOOK (ISBN, BOOK_NAME, PUBLISHER_CODE) VALUES (?, ?, ?)");
            stmt.setString(1, book.getIsbn());
            stmt.setString(2, book.getName());
            stmt.setString(3, book.getPublisher().getCode());
            stmt.executeUpdate();

            stmt.close();

            stmt = connection.prepareStatement("INSERT INTO CHAPTER (BOOK_ISBN, CHAPTER_NUM, TITLE) VALUES (?, ?, ?)");
            for (Chapter chapter: book.getChapters()) {
                stmt.setString(1, book.getIsbn());
                stmt.setInt(2, chapter.getChapterNumber());
                stmt.setString(3, chapter.getTitle());
                stmt.executeUpdate();
            }

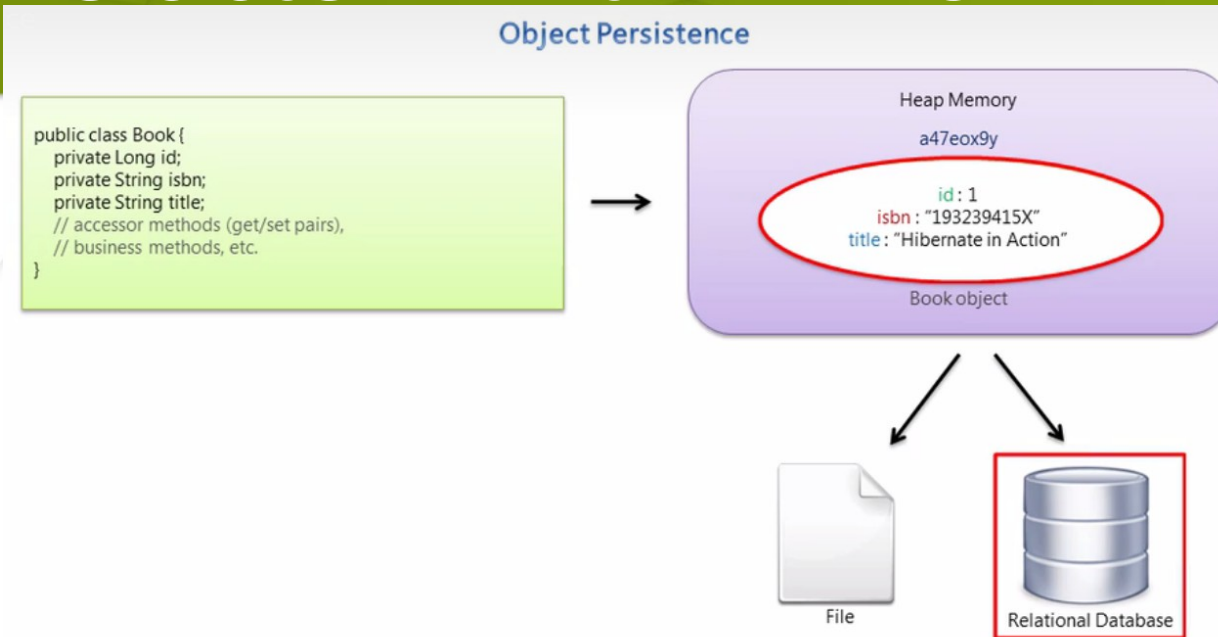
            stmt.close();
        } catch (ClassNotFoundException e) { e.printStackTrace(); } catch (SQLException e) { e.printStackTrace(); }
        finally { try { connection.close(); } catch (SQLException e) { e.printStackTrace(); } }
    }
}
```

- **JPA:**

- entityManager.persist(**publisher**);

- ....

# Persistenz mit RDBMS



- Objekte mit **Attributen** und **Methoden** (POJOs ) in relationalen Datenbanken speichern
- Objekte aus Datensätzen erzeugen
- Beziehungen zwischen Objekten auf Datenbank-Relationen abbilden

# Agenda

## Einstieg

ORM

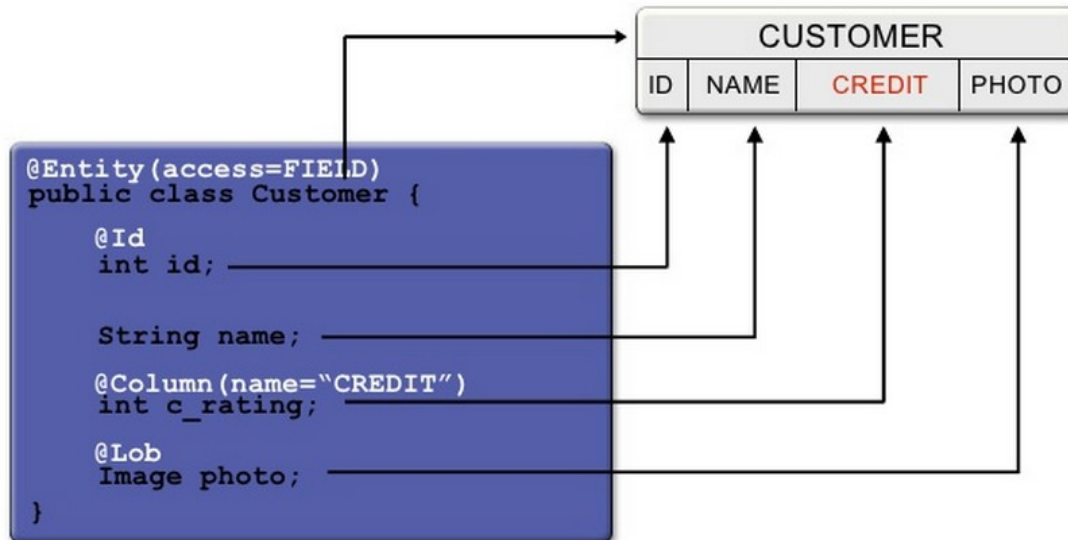
JPA

Hibernate

Andere Provider

# Ziele von JPA

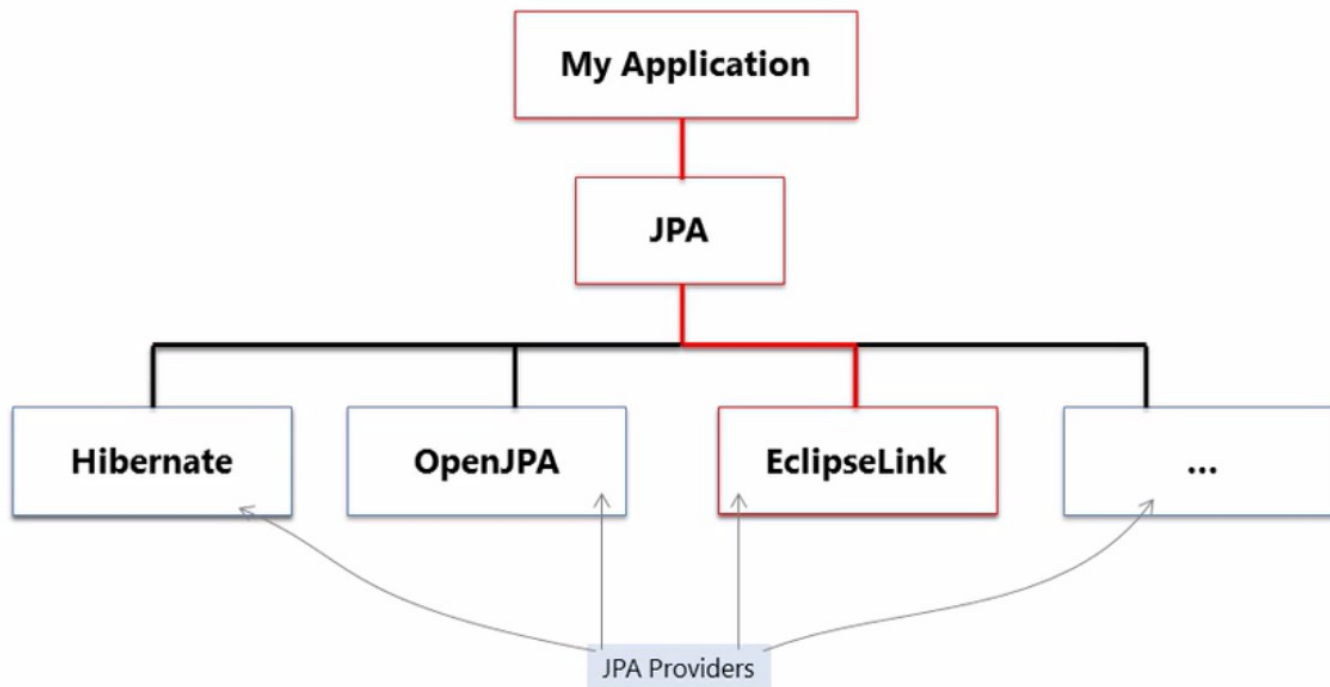
- Speicherung von Java Objekten (**POJOs**)
- Deklarative Konfiguration
- Kein natives SQL in den Klassen
- Transitive Persistenz von Abhängigkeiten



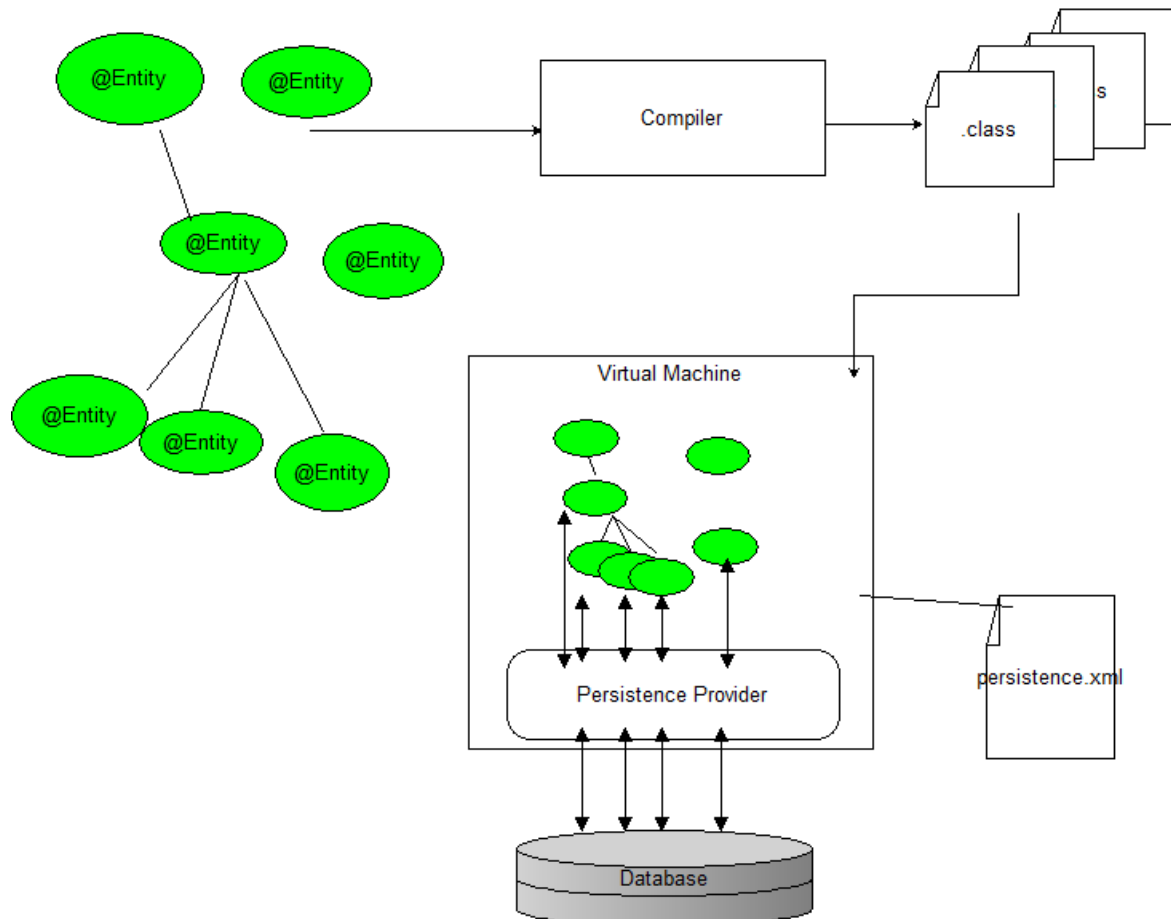


# Ziele von JPA

- Keine Abhängigkeit zu nativen Providern



# Wie funktioniert JPA (Annotationen)?



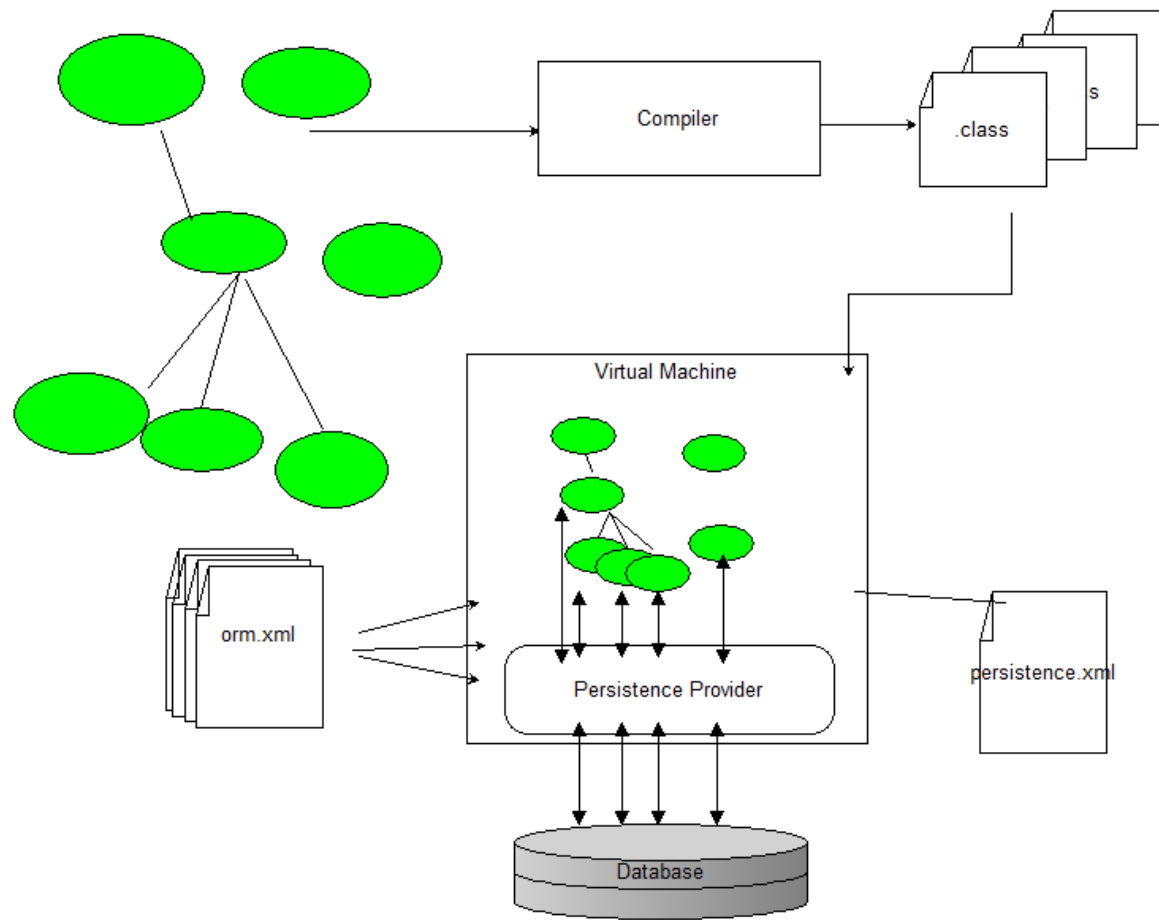
# Wie funktioniert JPA (XML Metadata)

ORM

JPA

Hibernate

Andere Provider



# Features

- Gutes Transaktionsmodell
- Transitive Persistenz von abhängigen Objekten
- POJO-basiertes Persistenzmodell
- Domainmodell unterstützt Vererbung, Polymorphie
- Anfragesprache
- Automatisches Dirty Checking
- Lazy Fetching
- Verfügbar in Java EE und Java SE
- Unabhängigkeit von Datenbank und OR-Mapper

# Managed / Unmanaged Environment

- JPA kann in zwei Laufzeitumfeldern benutzt werden
- **Unmanaged**
  - ◆ Explizites Connection- und Transaktionsmanagement

```
EntityManagerFactory emf =  
Persistence.createEntityManagerFactory("TestPU");  
EntityManager em = emf.createEntityManager();
```
- **Managed**
  - ◆ Impliziertes Connection- und Transaktionsmanagement

```
@PersistenceContext  
private EntityManager em;
```

# JPA - Java Persistence API

- keine Implementierung, nur Spezifikation
- wurde als Projekt der JSR 220 Expert Group entwickelt
- im Mai 2006 erstmals veröffentlicht
- aktuelle Version 2.1 wurde am 22. April 2013 freigegeben
- EclipseLink ist die Referenzimplementierung für die Java Persistence API (JPA) 2.1 und 2.0.
- TopLink Essentials war die Referenzimplementierung für JPA 1.0.

# JPA Provider



► <http://hibernate.org/>



► <http://www.eclipse.org/eclipselink/>



► <http://openjpa.apache.org/>



► <http://www.datanucleus.org/>

# Agenda

## **Einstieg**

ORM

JPA

**Hibernate**

Andere Provider



# Website

Einstieg

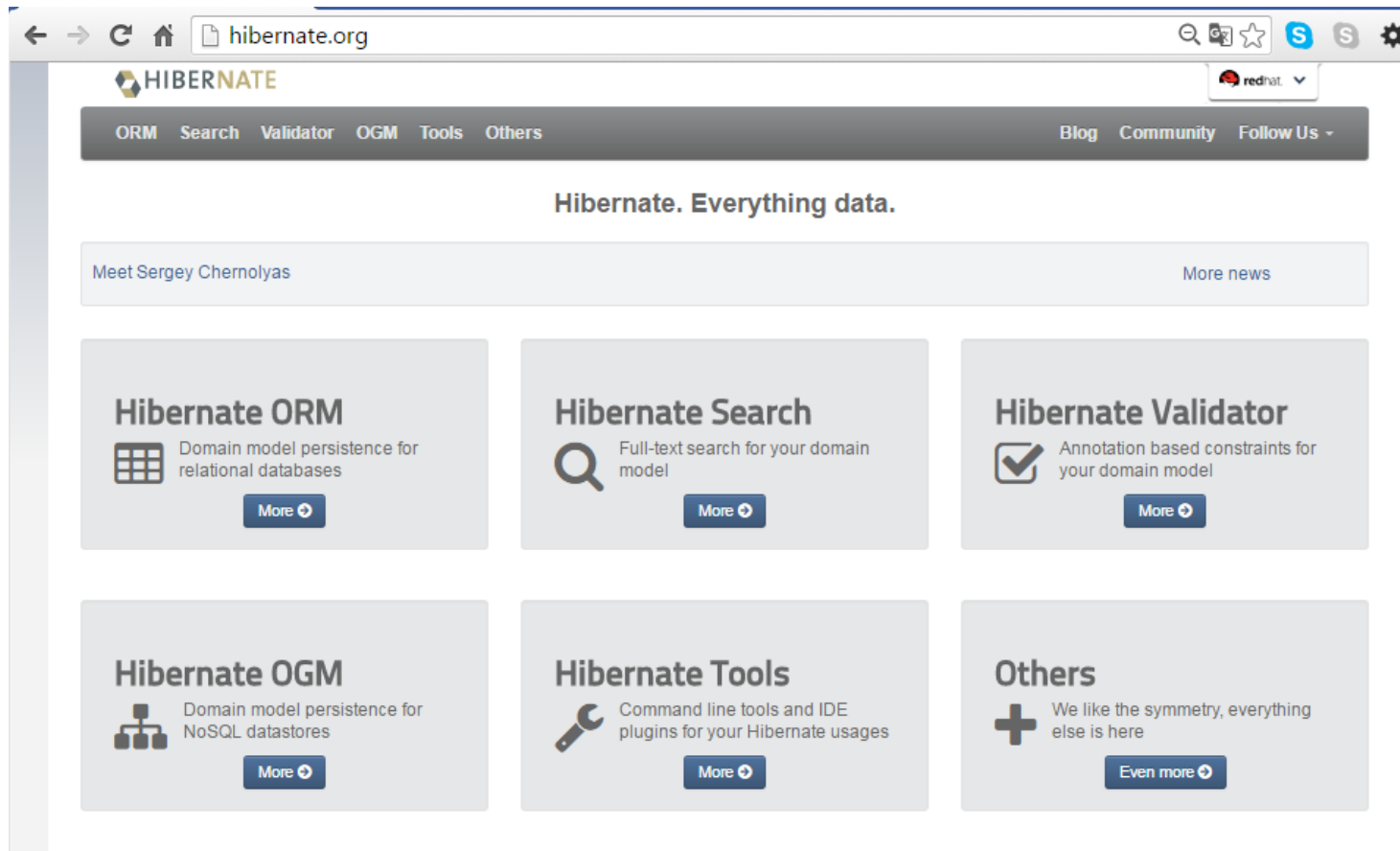
ORM

JPA

Hibernate

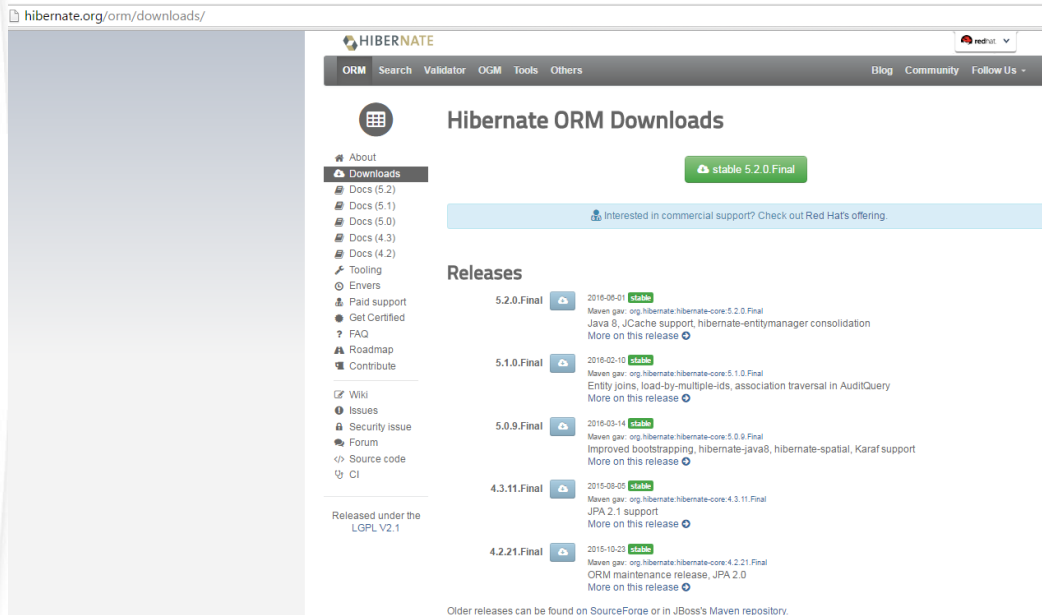
Andere Provider

- URL: <http://www.hibernate.org>



# Versionen

- URL: <http://www.hibernate.org>
- Versionen:
  - ◆ Version 3.x seit 2005
  - ◆ Version 4.x seit 2011
  - ◆ Version 5.x seit 2016



The screenshot shows the 'Hibernate ORM Downloads' page on the hibernate.org website. The page features a navigation menu on the left with links to About, Downloads, Docs (5.2, 5.1, 5.0, 4.3, 4.2), Tooling, Envers, Paid support, Get Certified, FAQ, Roadmap, and Contribute. The main content area displays the 'stable 5.2.0.Final' version prominently. Below this, a 'Releases' section lists several versions with their release dates and descriptions: 5.2.0.Final (2016-06-01), 5.1.0.Final (2016-02-10), 5.0.9.Final (2016-03-14), 4.3.11.Final (2015-08-05), and 4.2.21.Final (2015-10-23). Each release entry includes a 'stable' badge and a link to 'More on this release'. At the bottom, a note states: 'Older releases can be found on SourceForge or in JBoss's Maven repository.'

# Hibernate Module

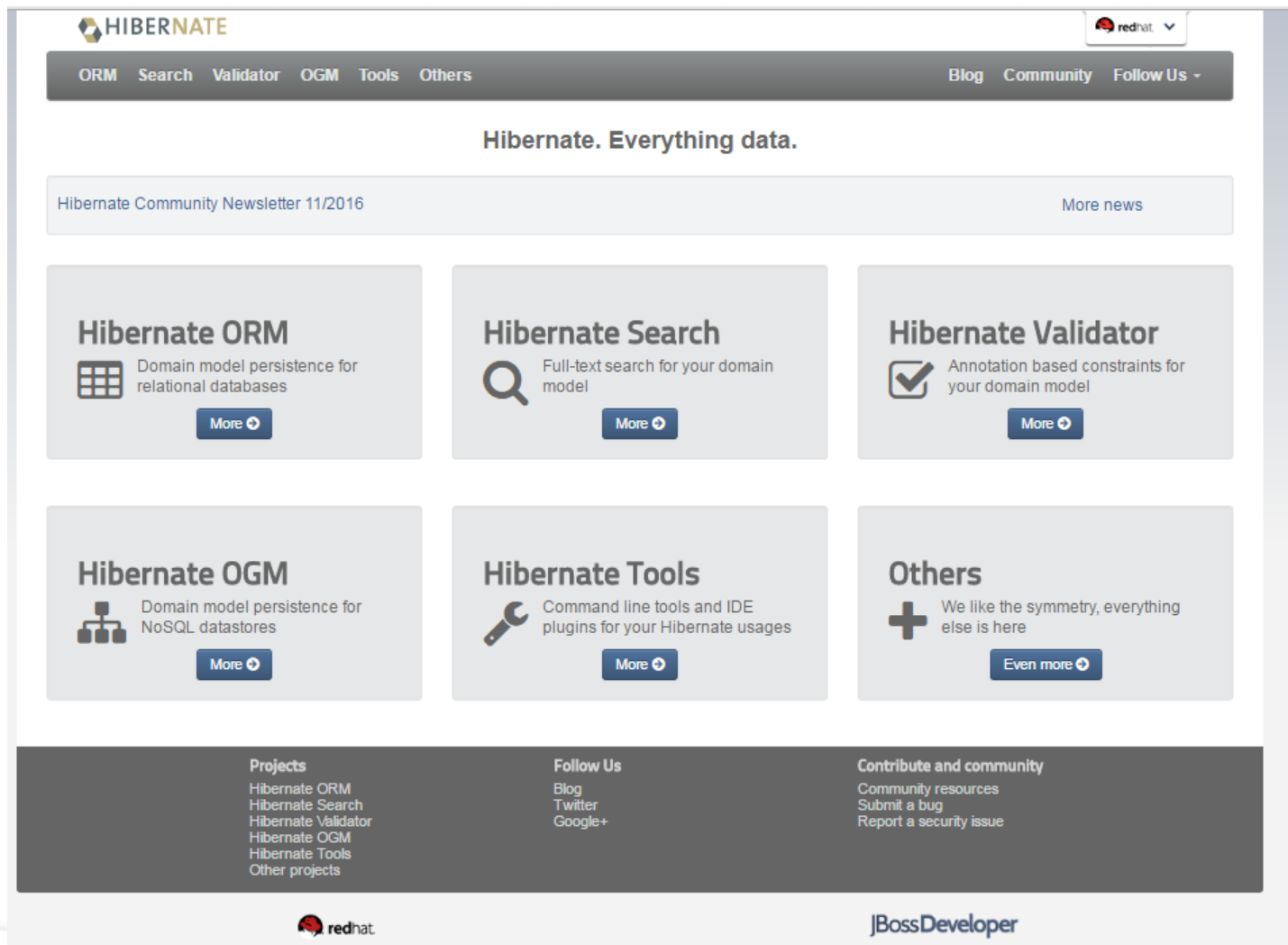
Einstieg

ORM

JPA

Hibernate

Andere Provider

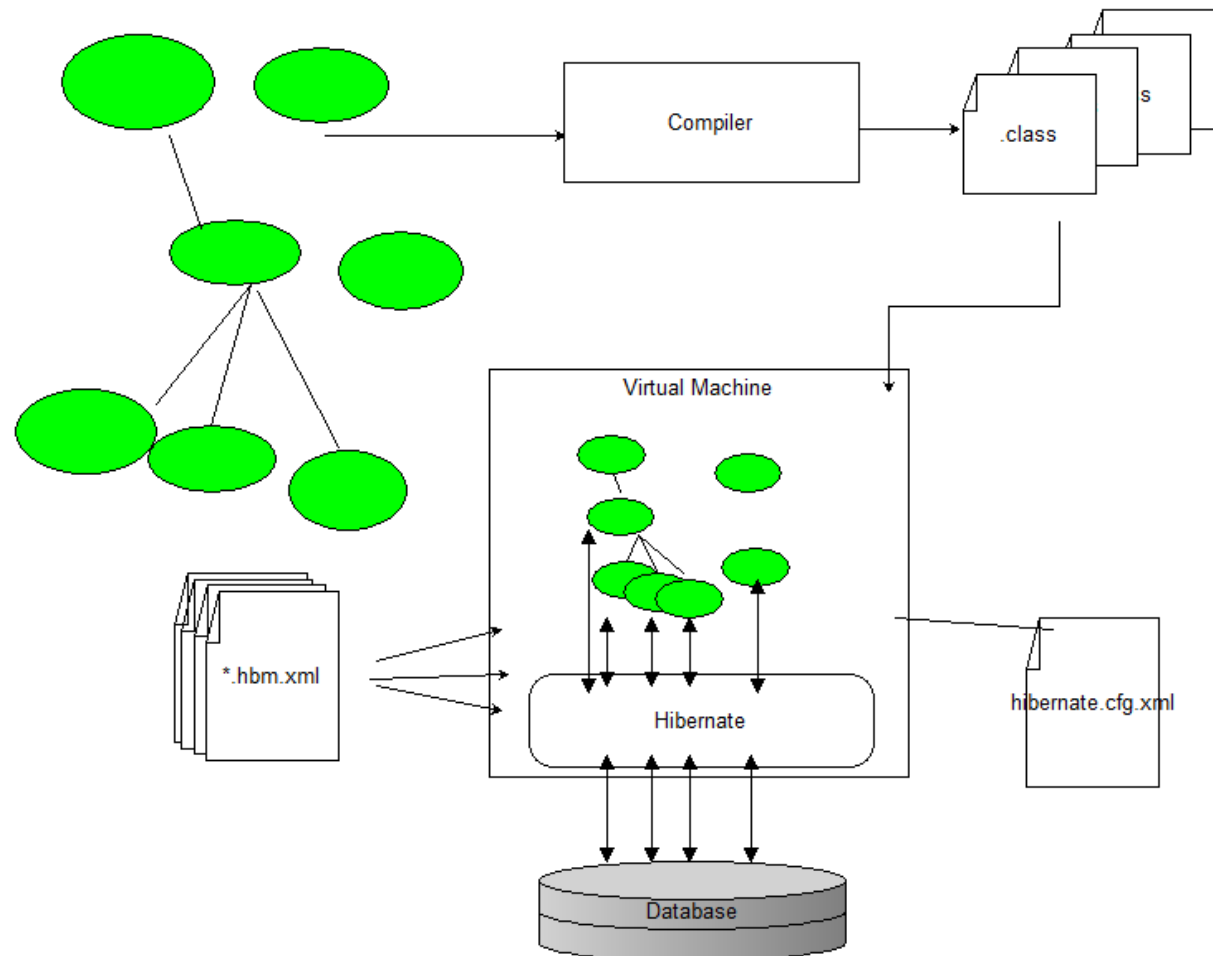


The screenshot shows the Hibernate website homepage. At the top, there's a navigation bar with links for ORM, Search, Validator, OGM, Tools, and Others. A redhat logo is in the top right corner. Below the navigation bar, the main heading reads "Hibernate. Everything data." followed by a link to the "Hibernate Community Newsletter 11/2016" and a "More news" link. The main content area is divided into six sections, each with an icon, a title, a brief description, and a "More" button:

- Hibernate ORM**: Domain model persistence for relational databases.
- Hibernate Search**: Full-text search for your domain model.
- Hibernate Validator**: Annotation based constraints for your domain model.
- Hibernate OGM**: Domain model persistence for NoSQL datastores.
- Hibernate Tools**: Command line tools and IDE plugins for your Hibernate usages.
- Others**: We like the symmetry, everything else is here.

The footer contains three columns of links: "Projects" (listing ORM, Search, Validator, OGM, Tools, and Other projects), "Follow Us" (listing Blog, Twitter, and Google+), and "Contribute and community" (listing Community resources, Submit a bug, and Report a security issue). The redhat logo and "JBossDeveloper" text are at the bottom center.

# Wie funktioniert Hibernate (XML Metadata Mapping)?



# Agenda

## **Einstieg**

ORM

JPA

Hibernate

**Andere Provider**

# Andere Provider

- EclipseLink 2.4.x
- <http://www.eclipse.org/eclipselink>



- OpenJPA 2.4.x
- <http://openjpa.apache.org/>

