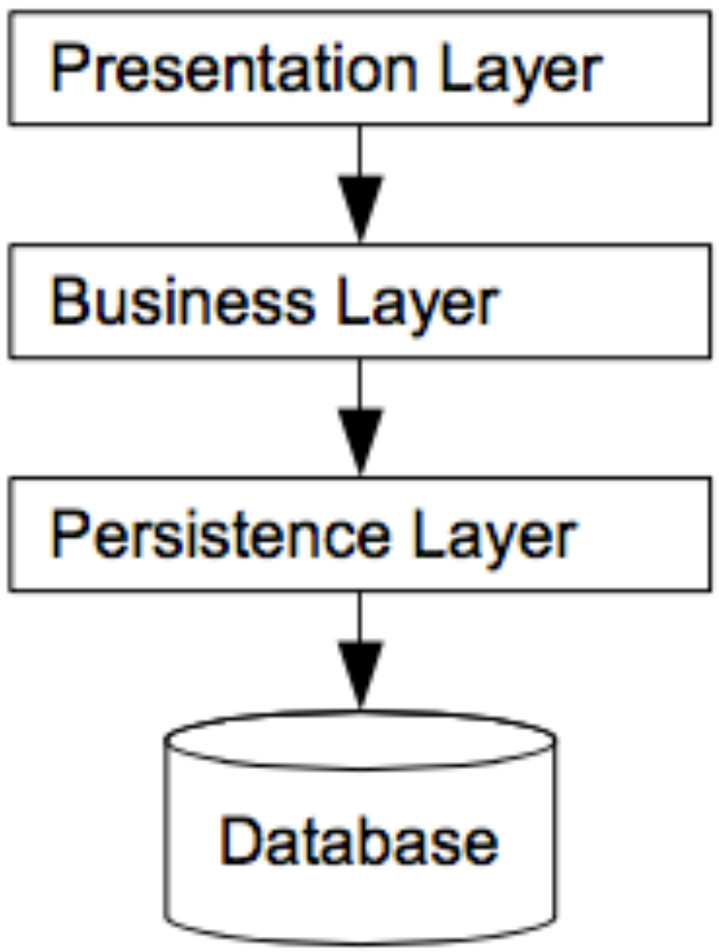
OBJECT-RELATIONAL MAPPING (ORM)

# JPA / HIBERNATE

# WAAROM?

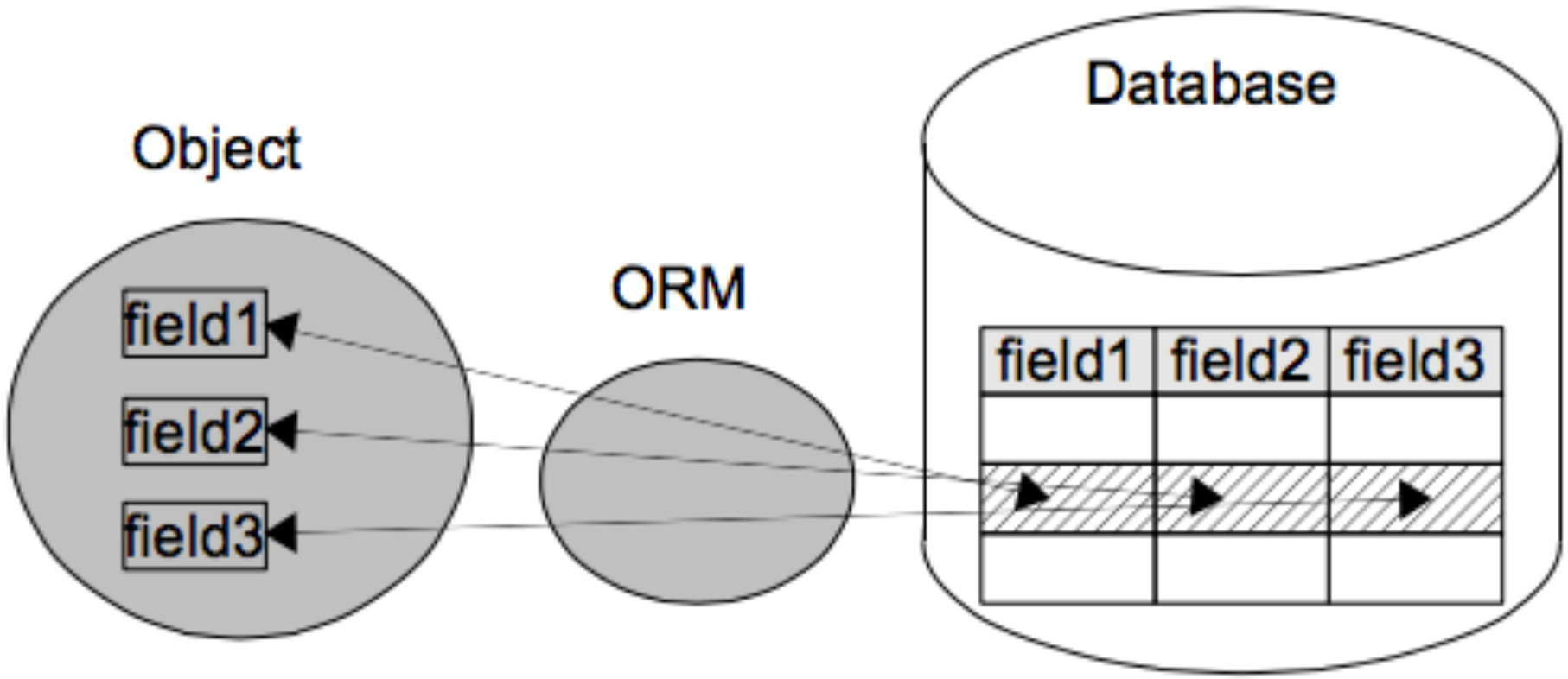
- gericht op relationele databanken
- JDBC: veel commando's nodig
  - snel verstrengelt met business logic
- ORM
  - separation of concerns : elke laag heeft zijn eigen verantwoordelijkheid

# ORM - LAYERED MODEL



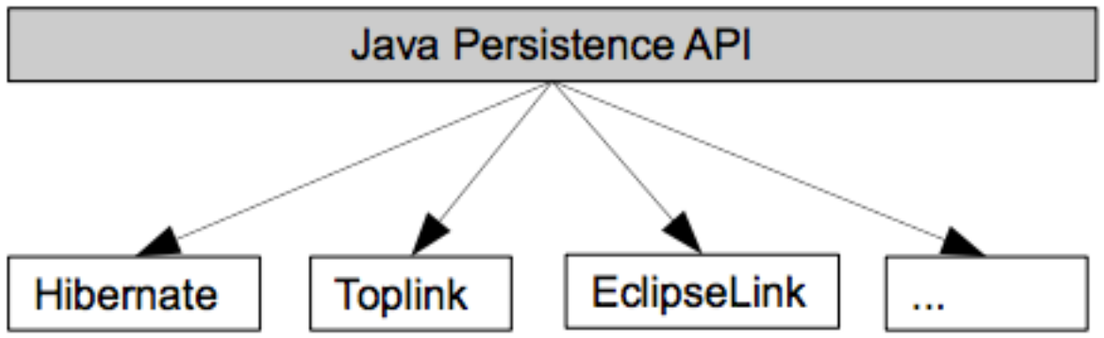

- enkel persistence layer communiceert met database (hier hoort de JDBC code thuis)
- vertaalt relationele tabellen in java objecten
- mapping tussen object-graph model en relationeel tabellen model

# ORM - MAPPING


Object
field1
field2
field3
ORM
Database
field1
field2
field3

# ORM - MAPPING

- EJB: enterprise java beans
  - verweven met application server
  - omslachtig
- Hibernate / EclipseLink(voorheen TopLink)
  - POJO principe




- Java Persistence API (JPA) 2.1
  - programmeermodel
  - verschillende implementaties, bvb Hibernate of EclipseLink
  - één api, onderling uitwisselbare implementaties

# ORM - INSTALLATIE (JPA MET HIBERNATE)

```
<dependencies>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-entitymanager</artifactId>
        <version>4.3.8.Final</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>[5,]</version>
    </dependency>
</dependencies>
```

# ORM - CONFIGURATIE - META-INF/PERSISTENCE.XML

- configuratie
  - database, username, password
  - mapping tussen tabellen en objecten
    - via annotaties
    - via META-INF/orm.xml (deployment descriptor)
      - perfecte scheiding
      - persistent maken van code zonder broncode bvb van andere library
- structuur volgens xml schema

```
<?xml version="1.0" encoding="UTF-8"?>
<entity-mappings version="2.1"
   xmlns="http://xmlns.jcp.org/xml/ns/persistence/orm"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence/orm
      http://xmlns.jcp.org/xml/ns/persistence/orm_2_1.xsd">

</entity-mappings>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
       http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
   version="2.1">
</persistence>
```

# ORM - HELLO WORLD!

- entity
  - volgen de javabean specificatie
  - @Entity
  - @Id

```
package messages;
import javax.persistence.*;

@Entity
public class Message {
   private long id;
   private String text;

   public Message() {
   }

   public Message(long id, String text) {
      this.id = id;
      this.text = text;
   }

   @Id
   public long getId() {
      return id;
   }

   public void setId(long id) {
      this.id = id;
   }

   public String getText() {
      return text;
   }

   public void setText(String text) {
      this.text = text;
   }
```

# ORM - HELLO WORLD!

- persistence-unit
  - geheel van classes die gebruik maken van 1zelfde database configuratie
  - discovery mbv annotation scanning

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
   version="2.1">
   <persistence-unit name="course"
                     transaction-type="RESOURCE_LOCAL">
      <properties>
         <property name="javax.persistence.jdbc.driver"
                value="com.mysql.jdbc.Driver" />
         <property name="javax.persistence.jdbc.url"
                value="jdbc:mysql://noelvaes.eu:3306/StudentDB" />
         <property name="javax.persistence.jdbc.user"
                value="student" />
         <property name="javax.persistence.jdbc.password"
                value="student123" />
         <property
         name="javax.persistence.schema-generation.database.action"
         value="create" />
         <!-- Hibernate specific -->
         <property name="hibernate.show_sql" value="true" />
      </properties>
   </persistence-unit>
</persistence>
```

Met de *property* `javax.persistence.schema-generation.database.action` kunnen we aangeven dat de tabellen automatisch gegenereerd en/of verwijderd worden door JPA. Mogelijke waarden zijn: `none`, `create`, `drop-and-create`, `drop`.

# ORM - HELLO WORLD!

```
package messages;
import javax.persistence.*;

public class SaveMessage {
   public static void main(String[] args) {
      EntityManagerFactory emf = Persistence
            .createEntityManagerFactory("course");
      EntityManager em = emf.createEntityManager();
      EntityTransaction tx = em.getTransaction();
      tx.begin();
      Message message = new Message(1, "Hello World");
      em.persist(message);
      tx.commit();
      em.close();
      emf.close();
      System.out.println("Message saved.");
   }
}
```

```
public class GetMessage {
   public static void main(String[] args) {
      EntityManagerFactory emf = Persistence
            .createEntityManagerFactory("course");
      EntityManager em = emf.createEntityManager();
      EntityTransaction tx = em.getTransaction();
      tx.begin();
      Message  message = em.find(Message.class,1L);
      System.out.println(message.getText());
      tx.commit();
      em.close();
      emf.close();
   }
}
```