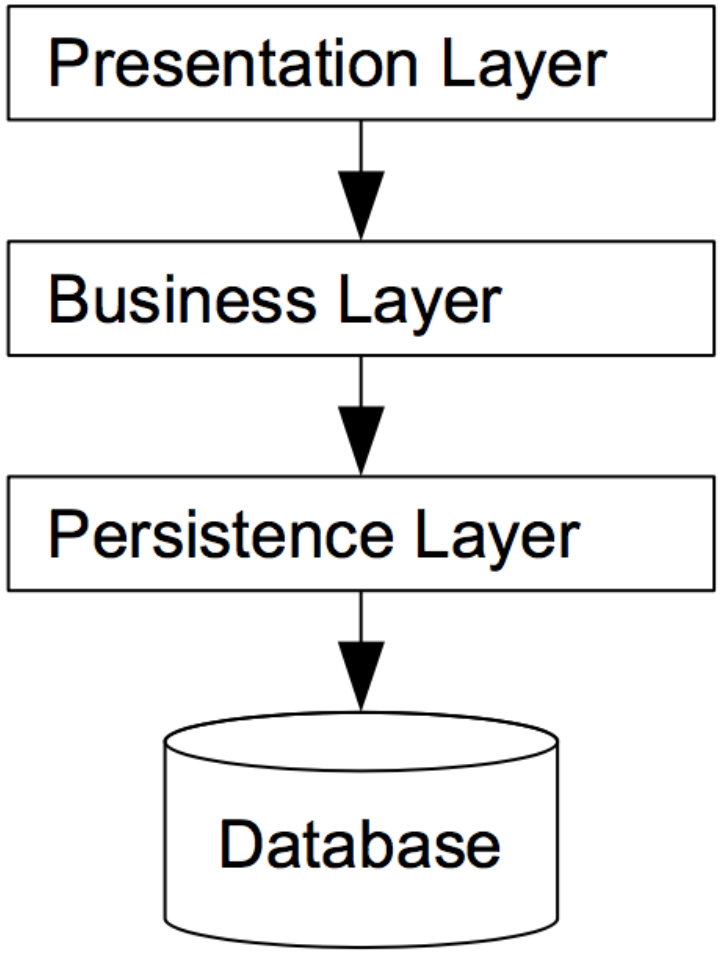
Programming Advanced java

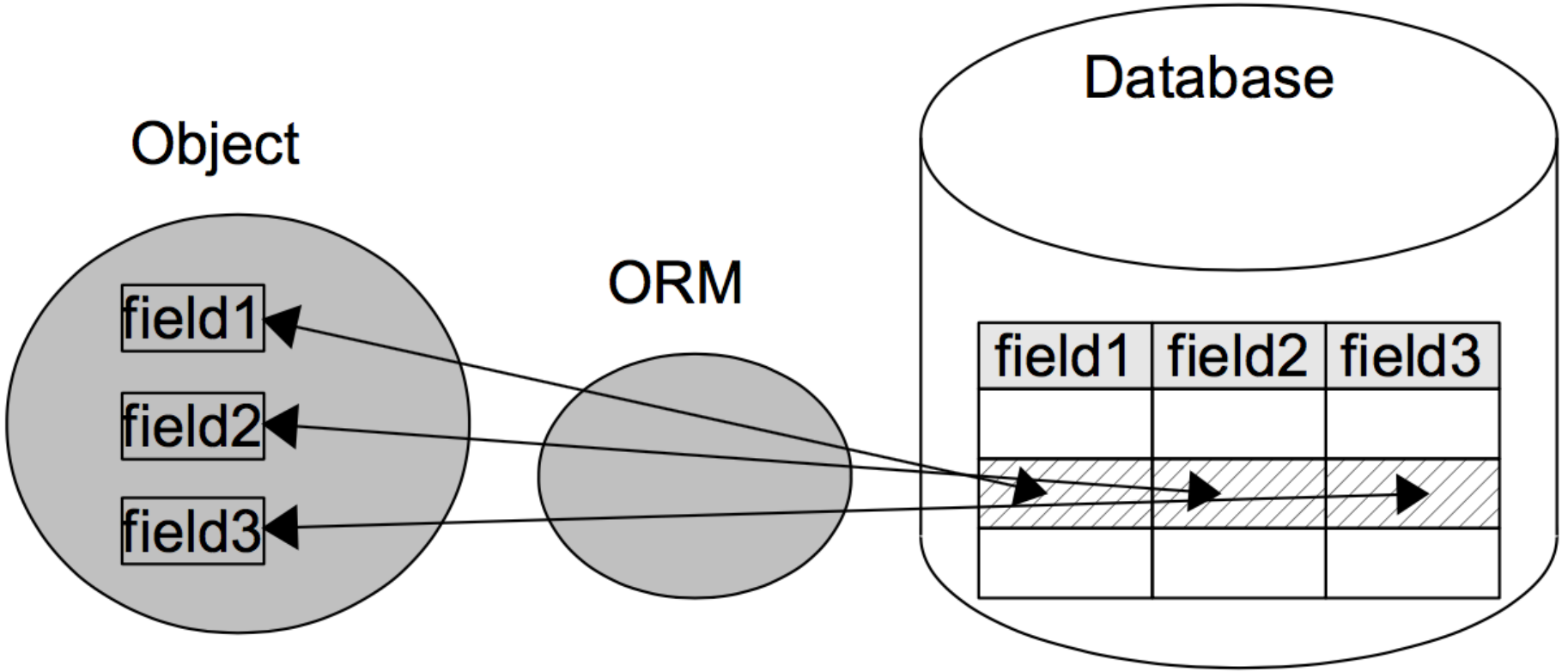
JPA

HOGESCHOOL PXL

Layer architecture



Object Relation Mapping (ORM)



Object Relation Mapping (ORM)

DAO:

- Veel boilerplate code
- Veel maintenance

→Enterprise JavaBeans 2.0

- Entity Beans
- Runtime container nodig
- Application Server
- Log en omslachtig in gebruik

→ POJO → Hibernate / TopLink

→Enterpise JavaBeans 3.0

→Java Persistence API (JPA)

→persistence layer gebruik makend van POJO's

JPA

→Enterpise JavaBeans 3.0

→Java Persistence API (JPA)

- persistence layer gebruikmakend van POJO's
- specificatie
- implementaties
  - Hibernate
  - TopLink
  - EclipseLink
  - OpenJPA
  - …

→ JPA 2.1 met Hibernate als implementatie

JPA - Hibernate

```
<dependencies>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-entitymanager</artifactId>
        <version>5.2.12.Final</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.45</version>
    </dependency>
</dependencies>
```

JPA – Hibernate - Configuratie

- **Database configuratie**
  → META-INF/persistence.xml
    - databaseschema
    - username
    - password

- **Mapping configuratie** (relationele tabellen <-> java-objecten)

  → Deployment descriptor
    - META-INF/orm.xml
    - scheiding code en configuratie
    - bestaande klassen gebruiken zonder code aan te passen

  → Annotations
    - makkelijk en snel in gebruik
    - java code

persistence.xml

```
<?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="example">
    <properties>
      <!-- JPA standard -->
      <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
      <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost/Example" />
      <property name="javax.persistence.jdbc.user" value="..." />
      <property name="javax.persistence.jdbc.password" value="..." />

      <property name="javax.persistence.schema-generation.database.action" value="drop-and-create" />

      <!-- Hibernate specific -->
      <property name="hibernate.show_sql" value="true"/>
      <property name="hibernate.xxx" value="xxx"/>
    </properties>
  </persistence-unit>
</persistence>
```

Entity class

```
@Entity
public class Message {

   @Id
   private long id;
   private String text;

   public Message() {
   }

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

 // getters and setters
 …

}
```

Primary keys - datatypes

- primitive (byte, int, short, long, char, float, double)
- primitive wrapper-class
- String
- Big numeric type (BigInteger)
- Date

Primary keys - type

- Natuurlijke primary key
  - Key met betekenis in de value
  - vb. rijksregisternummer
  - Indien mogelijk vermijden

- Surrogaat primary key
  - Oplopende nummer
  - Geen logische betekenis
  1. Enkelvoudig (@Id)
  2. Samengesteld (@IdClass)
  3. Autogenerated (@GeneratedValue)

Primary keys - autogenerated

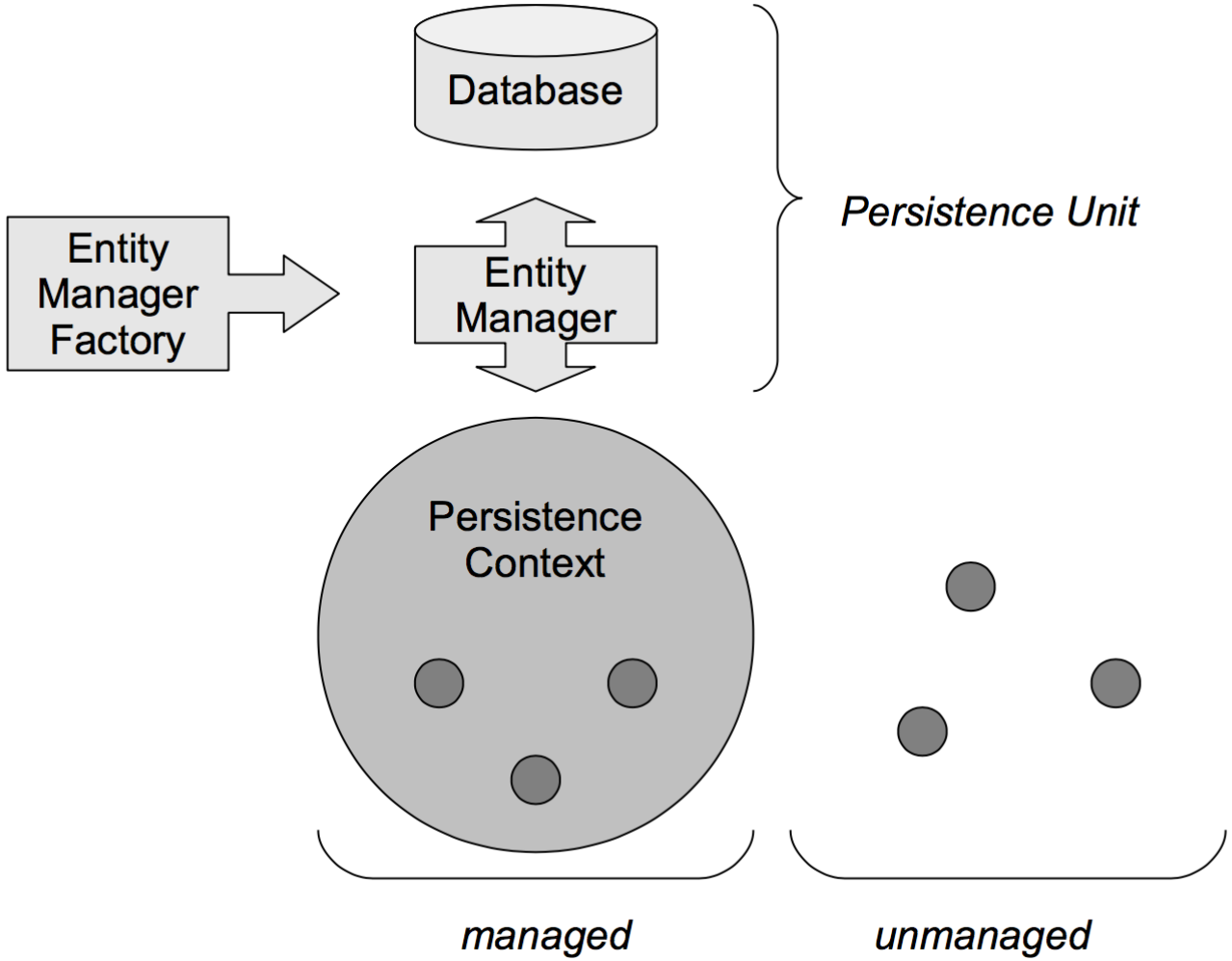
@GeneratedValue(strategy=GenerationType.AUTO)

Strategie	***Omschrijving***
`IDENTITY`	Er wordt gebruikgemaakt van een *identity column* in de databank. Dit is een kolom waarvan de inhoud automatisch ophoogt bij de creatie van een record (*auto-increment*).
`SEQUENCE`	Er wordt gebruikgemaakt van een *sequence* in de databank.
`TABLE`	Er wordt gebruikgemaakt van een aparte tabel waarin telkens de hoogste waarde van de *primary key* bewaard wordt.
`AUTO`	De JPA-*provider* kiest zelf de beste strategie, rekening houdend met de mogelijkheden van de onderliggende databank. Dit is tevens de standaardwaarde.

Entity usage

```
EntityManagerFactory entityManagerFactory =
Persistence.createEntityManagerFactory("example");

EntityManager entityManager =
entityManagerFactory.createEntityManager();

EntityTransaction transaction =
entityManager.getTransaction();

transaction.begin();
Message message = new Message(1, "Hello Hibernate");
entityManager.persist(message);
transaction.commit();

entityManager.close();
entityManagerFactory.close();
```

Persistence Context



Persistence Context

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

public class ChangeMessage {
   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);
      message.setText("Hello Mars");  // managed
      tx.commit();
      em.close();
      message.setText("Hello Venus"); // unmanaged
      emf.close();
   }
}
```

JPA Transactions

- Alle wijzigingen aan een object binnen een persistence context kunnen enkel via een transactie naar de databank weggeschreven worden
  - transaction.begin()
  - transaction.commit()
  - transaction.rollback()
- Wijzigingen binnen een persistence context maar buiten een transactie worden pas in rekening gebracht bij de eestvolgende voltooiing van een transactie

JPA Transactions - flush

- Wijzigingen doorvoeren binnen een transactie alvorens de commit
- FlushMode
  - FlushModeType.AUTO
    - Wegschrijven na commit
    - Wegschrijven net voor zoekoperatie
    - Default
  - FlushModeType.COMMIT
    - Wegschrijven enkel na commit
    - Onnodige databaseoperaties worden vermeden

Entity Manager

`persist`	Toevoegen van een object aan de persistence context Wegschrijven naar de database is afhankelijk van de implementatie. → Rekening houden met automatisch gegenereerde primary keys
`find`	Opzoeken record en toevoegen aan de persistence context
`merge`	Een unmanaged object samenvoegen met een managed object
`remove`	Verwijderen van een object uit de persistence context en database
`refresh`	Gegevens van een managed object opnieuw uitlezen uit de database
`clear`	Leegmaken van de persistence context.
`detach`	Een managed object unmanaged maken
`contains`	Nagaan of een object zicht in de persistence context bevindt