Hibernate Introduction

Maven, Hibernate, Configuration, JPA, Annotations

SoftUni Team

Technical Trainers

Software University

SoftUni

sli.do

#Java-DB

Table of Contents

1. Maven

2. Hibernate Framework

3. Java Persistence API

Maven

Project Management and Comprehension
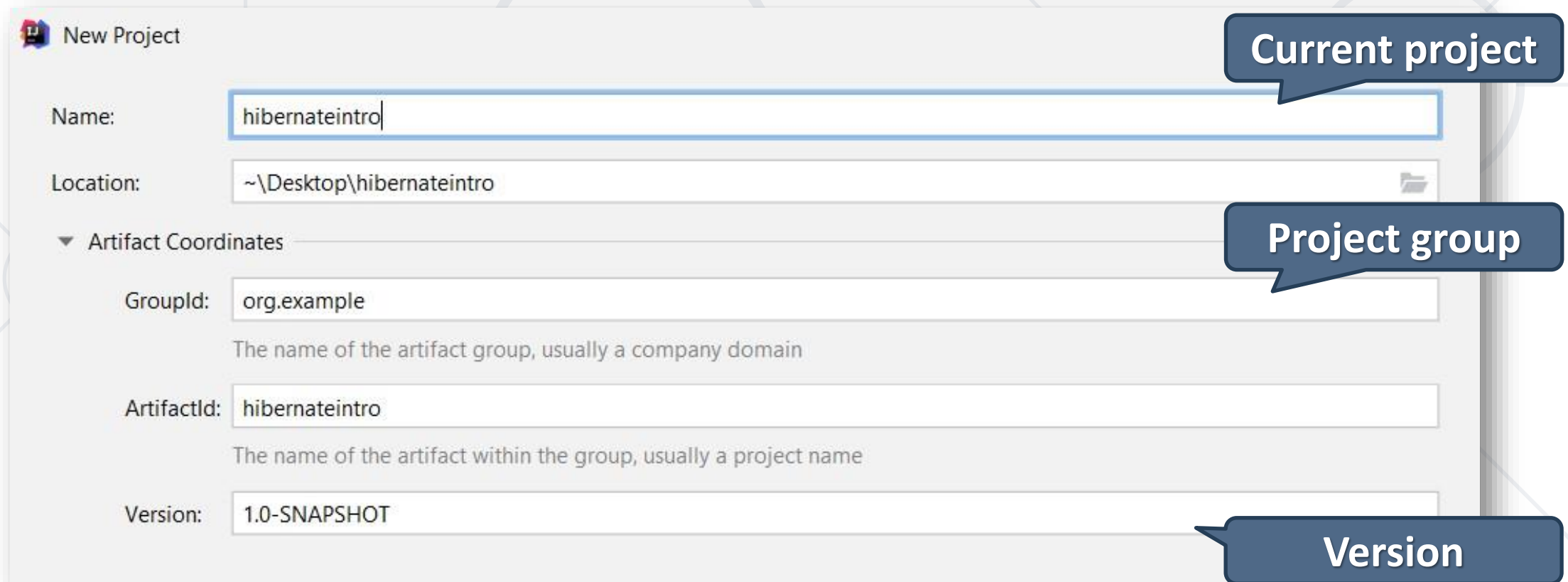
Maven Overview

- Maven is a built automation tool.

    - Describes how software is built and its dependencies

    - Uses XML files

- Dynamically downloads **Java libraries** and **Maven plug-ins**

    - Projects are configured using a **P**roject **O**bject **M**odel, which is stored in a **pom.xml** file

Setup – Creating a Maven Project

- Select "Maven" project from the new project panel:
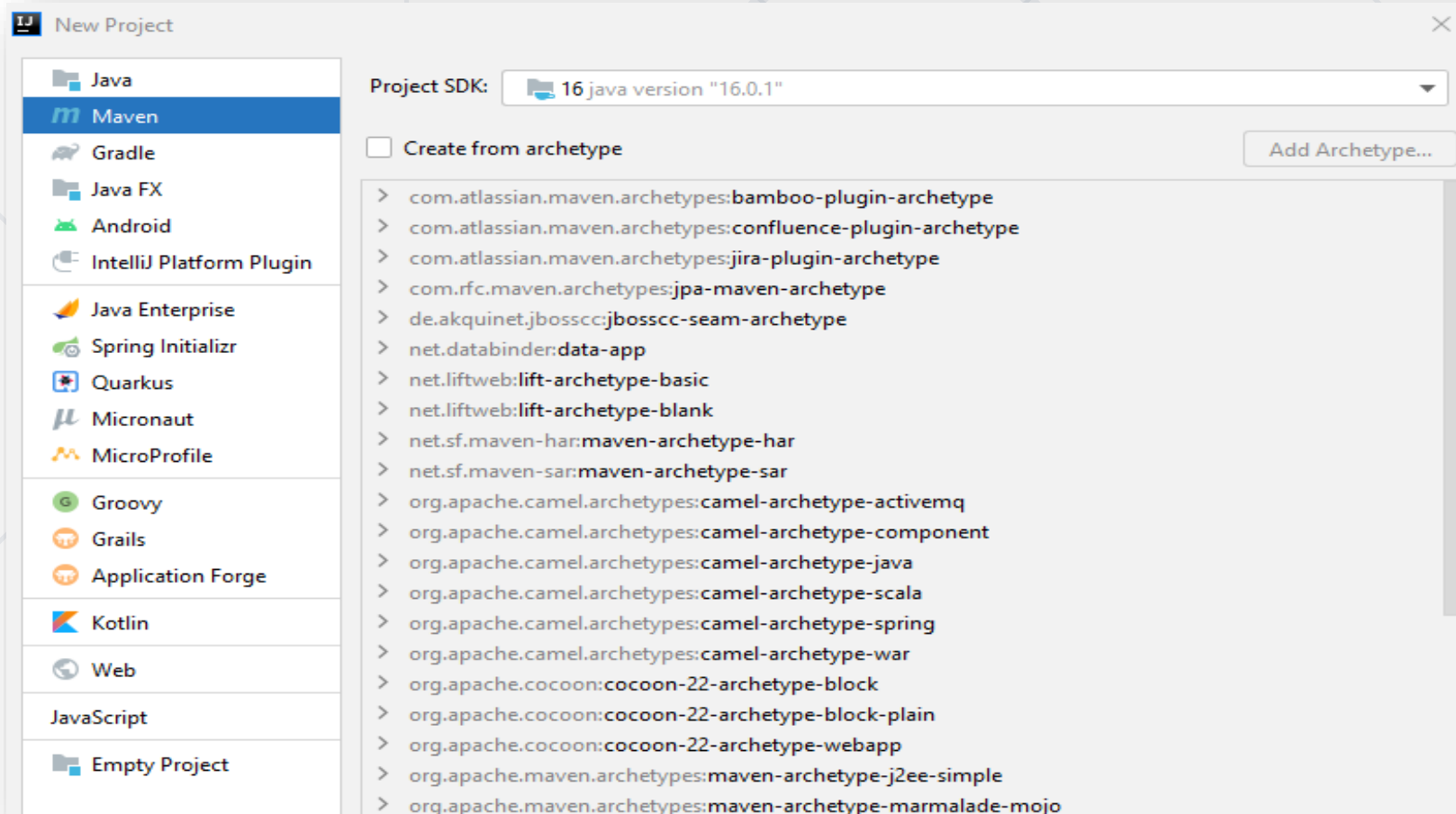
Setup

Current project

Project group

Version

Maven Configurations

- A **P**roject **O**bject **M**odel(**POM**) is the fundamental unit of work in Maven

- **Configurations** are held in the **pom.xml** file

  - When executing a task or goal, Maven looks for the POM file in the current directory

POM Model

```
pom.xml

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
                <source>16</source>
                <target>16</target>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Java compile version

Dependencies

- Dependencies are set with the **<dependency>** tag:

```
pom.xml

<dependencies>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>
</dependencies>
```

Dependency 1

Dependency 2

Hibernate Framework

Mapping Java Classes to Database Tables

Hibernate Framework

- Hibernate is a Java ORM framework

  - Mapping an object-oriented model to a relational database

    - It is implemented by the configuration of an **XML file** or by using **Java Annotations**

  - Maintain the database schema

Java ORM Approaches

- Different approaches to **Java ORM**:

  - POJO (Plain Old Java Objects) + XML mappings

    - A bit old-fashioned, but very powerful

    - Implemented in the "classical" Hibernate

  - Annotated Java classes (**POJO**) mapped to DB tables

    - Based on Java annotations and XML

    - Easier to implement and maintain

  - Code generation - tools

Hibernate Configuration

- Pom.xml explain

```
                                    pom.xml

        …
          <dependencies>
               <dependency>
                    <groupId>org.hibernate</groupId>
                    <artifactId>hibernate-core</artifactId>        Hibernate
                    <version>5.4.30.Final</version>
               </dependency>
               <dependency>
                    <groupId>mysql</groupId>
                    <artifactId>mysql-connector-java</artifactId>
               </dependency>                                MySQL connector
          </dependencies>
        …
```
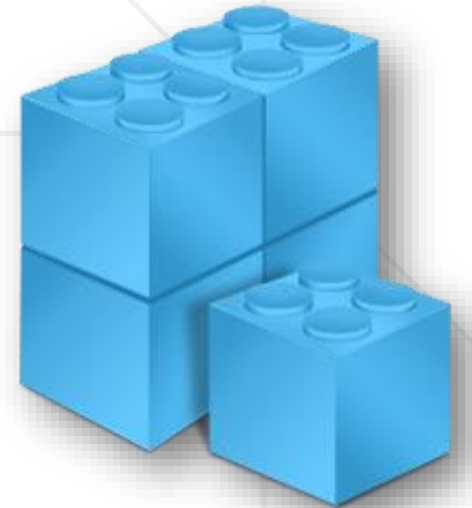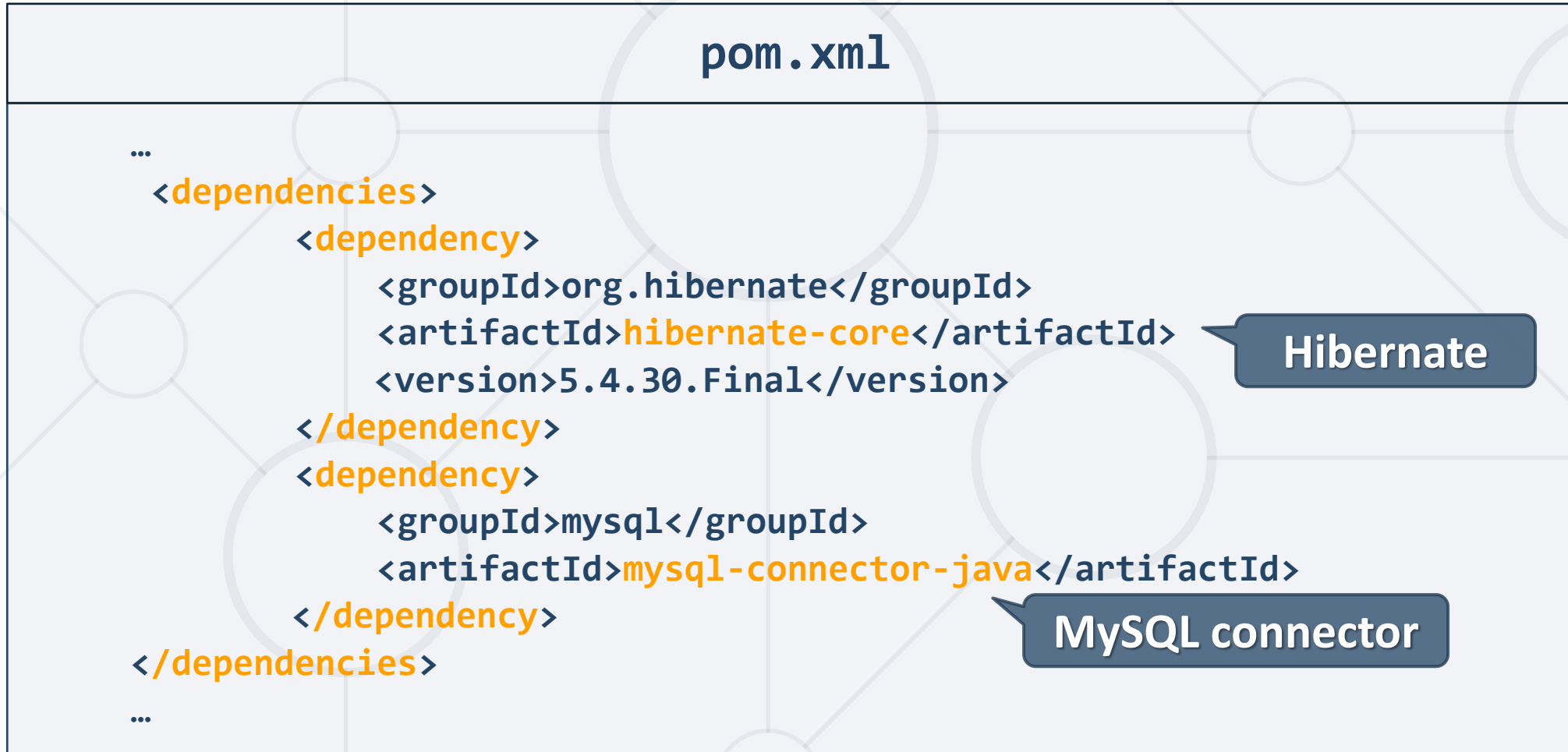
Hibernate Configuration

hibernate.cfg.xml

```xml
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration
        PUBLIC "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <property name="hibernate.dialect">
            org.hibernate.dialect.MySQL8Dialect
        </property>
        <property name="hibernate.connection.driver_class">
            com.mysql.cj.jdbc.Driver
        </property>
```

Configuration

SQL Dialect

Driver

hibernate.cfg.xml

```xml
<!-- Connection Settings -->
<property name="hibernate.connection.url">
jdbc:mysql://localhost:3306/school?createDatabaseIfNotExist=true
</property>
<property name="hibernate.connection.username">
    root
</property>
<property name="hibernate.connection.password">
    1234
</property>
<property name="hbm2ddl.auto">
    update
</property>
```

Connection string

User

Pass

Auto strategy

Hibernate Configuration

```
                          hibernate.cfg.xml
    …
        <!-- List of XML mapping files -->
        <mapping resource="student.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

Mapping files

Hibernate Implementation Example

- **POJO** (**P**lain **O**ld **J**ava **O**bjects) + XML mappings

```java
public class Student {

    private long id;

    private String name;

    public Student() {

    }
    // Getters and setters

}
```

Hibernate Mapping

student.hbm.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD//EN"
        "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
    <class name="entities.Student" table="students">
        <id name="id" column="id">
            <generator class="identity" />
        </id>
    …
```

Mapping file

Class mapping

Field mapping

Hibernate Mapping

```
student.hbm.xml
```
```
…
    <property name="name" column="first_name" />
  </class>
</hibernate-mapping>
```

Field mapping

Hibernate Sessions

Main.java

```java
public class Main {
    public static void main(String[] args) {
        Configuration cfg = new Configuration();
        cfg.configure();
        SessionFactory sessionFactory =
                cfg.buildSessionFactory();
        Session session = sessionFactory.openSession();
        session.beginTransaction();

        // Your Code Here
        session.getTransaction().commit();
        session.close();
    }
}
```

Service Registry

Session

Transaction commit

Hibernate Save Data

```
                        Main.java

public static void main(String[] args) {
    //…
    session.beginTransaction();

    Student example = new Student();
    session.save(example);

    session.getTransaction().commit();
        session.close();
    }
}
```

Save object

Hibernate Retrieve Data by Get

Main.java
| ```java
public static void main(String[] args) {

    …


    Student student = session.get(Student.class, 1L);


        session.close();
    }
}
``` |

Get object

Hibernate Retrieve Data by Query

Main.java

```java
public static void main(String[] args) {
    // …
        session.beginTransaction();

    List<Student> studentList =
    session.createQuery("FROM Student " ,
                        Student.class).list();
        for (Student student : studentList) {
            System.out.println(student.getId());
        }
        session.getTransaction().commit();
        session.close();
    }
}
```

Get list of objects

Hibernate Querying Language – HQL

SELECT

`"FROM Student"`

SELECT + WHERE

`"FROM Student WHERE name = 'John'"`

SELECT + JOIN

`"FROM Student AS s`
`JOIN s.major AS major"`

Hibernate Retrieve Data by Criteria

Main.java

```java
public static void main(String[] args) {
    //…
    session.beginTransaction();
    CriteriaBuilder builder = session.getCriteriaBuilder();
    CriteriaQuery criteria = builder.createQuery();
    Root<Student> r = criteria.from(Student.class);
    criteria.select(r).where(builder.like(r.get("name"),"P%"));
    List<Student> studentList =
    session.createQuery(criteria).getResultList();
        for (Student student : studentList) {
            System.out.println(student.getName());
        }


    session.getTransaction().commit();
    session.close();
}
```

Get list of objects by criteria

Java Persistence API

ORM Fundamentals

About JPA

- What is Java Persistence API (JPA)?
  - Database persistence technology for Java (**official standard**)
    - Object-relational mapping (ORM) technology
    - Operates with POJO entities with annotations or XML mappings
    - Implemented by many **ORM engines**: **Hibernate**, **EclipseLink**, etc.

About JPA

- JPA maps Java classes to database tables

  - Maps relationships between tables as associations between classes

- Provides **CRUD** functionality and queries

  - Create, read, update, delete + queries

Entities in JPA
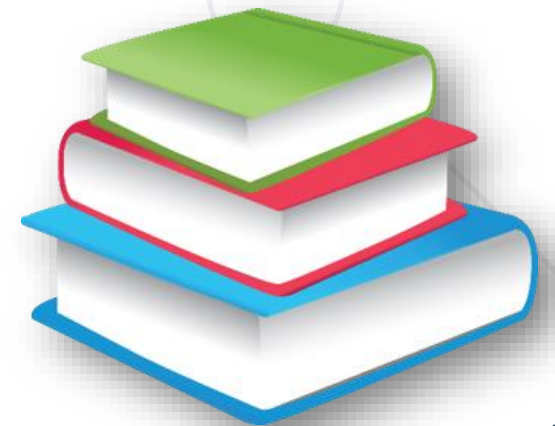
- A JPA entity is just a POJO class

  - Abstract or concrete **top level** Java class

  - Non-final fields/properties, no-arguments constructor

  - No required interfaces

  - Direct field or property-based access

- Getter/setter can contain logic (e.g., validation)

Entity Class: Student

```java
Student.java

@Entity @Table(name = "students")        Set table name
public class Student {
    @Id      Primary key                               Identity
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")       Column name
    private long id;
    @Column(name = "name", length = 50)        Column name
    private String name;                                and length
                                 Column name

    // Getters and setters

}
```

Annotations

- **@Entity** - Declares the class as an entity or a table

- **@Table** - Declares table name

- **@Basic** - Specifies non-constraint fields explicitly

- **@Transient** - Specifies the property that is not persistent, i.e., the value is never stored in the database

Annotations

- **@Id** - Specifies the property, use for identity (primary key of a table) of the class

  - **@GeneratedValue** - specifies how the identity attribute can be initialized

    - Automatic, manual, or value taken from a sequence table

- **@Column** -Specifies the column attribute for the persistence property

JPA Configuration

pom.xml

``xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 	
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
        <groupId>com.javawebtutor</groupId>
        <artifactId>JPAMavenExample</artifactId>
        <packaging>jar</packaging>
        <version>1.0-SNAPSHOT</version>
        <name>JPAMavenExample</name>
        <url>http://maven.apache.org</url>
    …
``

JPA Configuration

```xml
pom.xml

...
<dependencies>
        <dependency>
                <groupId>javax.persistence</groupId>
                <artifactId>javax.persistence-api</artifactId>
                <version>2.2</version>
        </dependency>
        <dependency>
                <groupId>org.hibernate</groupId>
                <artifactId>hibernate-core</artifactId>
                <version>5.4.30.Final</version>
        </dependency>
...
```

pom.xml

```xml
...
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.25</version>
    </dependency>
    </dependencies>
</project>
```

JPA Configuration

- Create new directory **META-INF** in **resources** folder. After that place persistence.xml in it

persistence.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence" version="2.0">
    <persistence-unit name="school">
        <properties> <property name = "hibernate.connection.url"
value="jdbc:mysql://localhost:3306/school?createDatabaseIfNotExist=true"/>
            <property name = "hibernate.connection.driver_class"
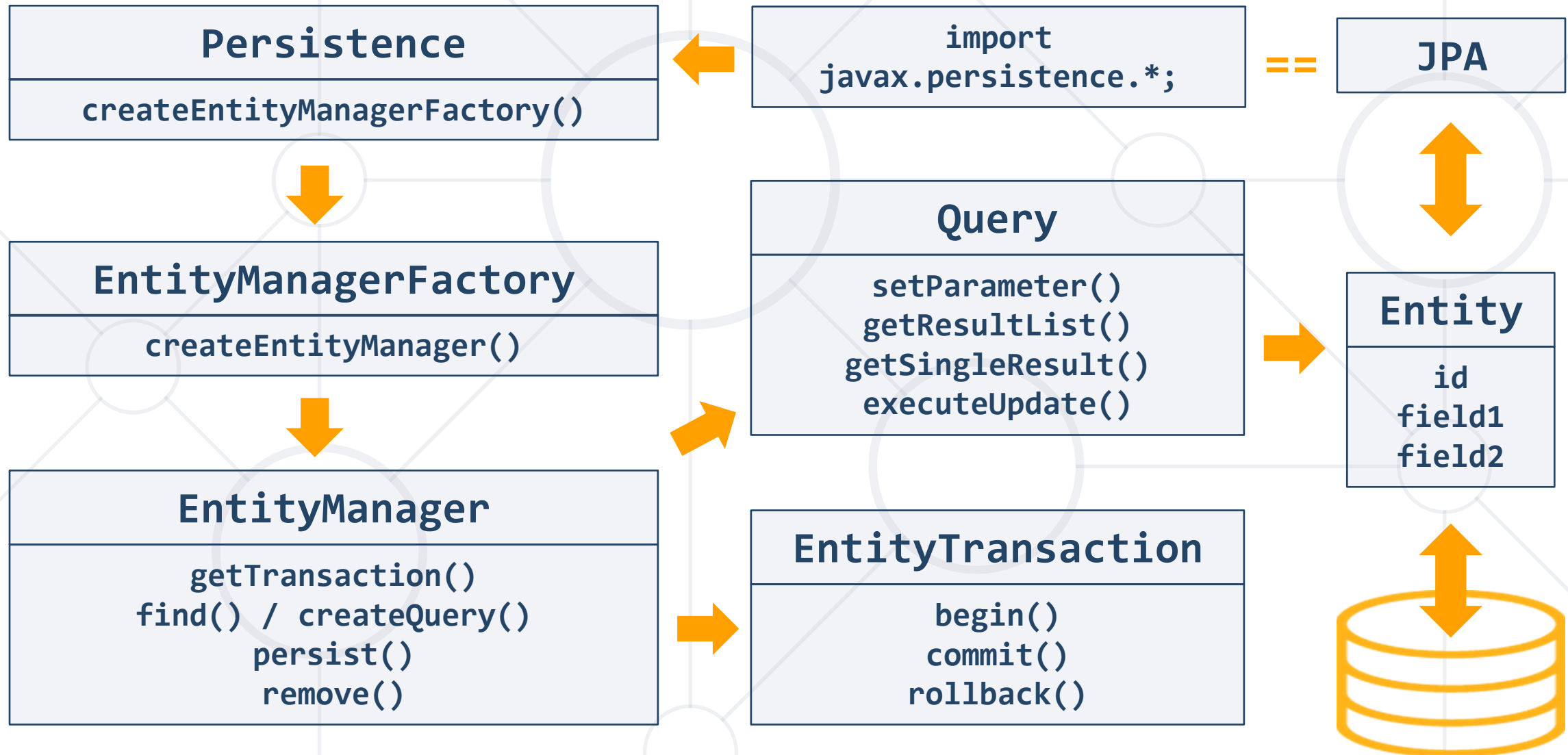value="com.mysql.jdbc.Driver"/>
        …
```

JPA Configuration

```
persistence.xml

        …
        <property name = "hibernate.connection.username" value="root"/>
        <property name = "hibernate.connection.password" value="1234"/>
        <property name = "hibernate.dialect"
value="org.hibernate.dialect.MySQL8Dialect"/>
        <property name = "hibernate.hbm2ddl.auto" value="update"/>
        <property name = "hibernate.show_sql" value = "true" />
      </properties>
    </persistence-unit>
</persistence>
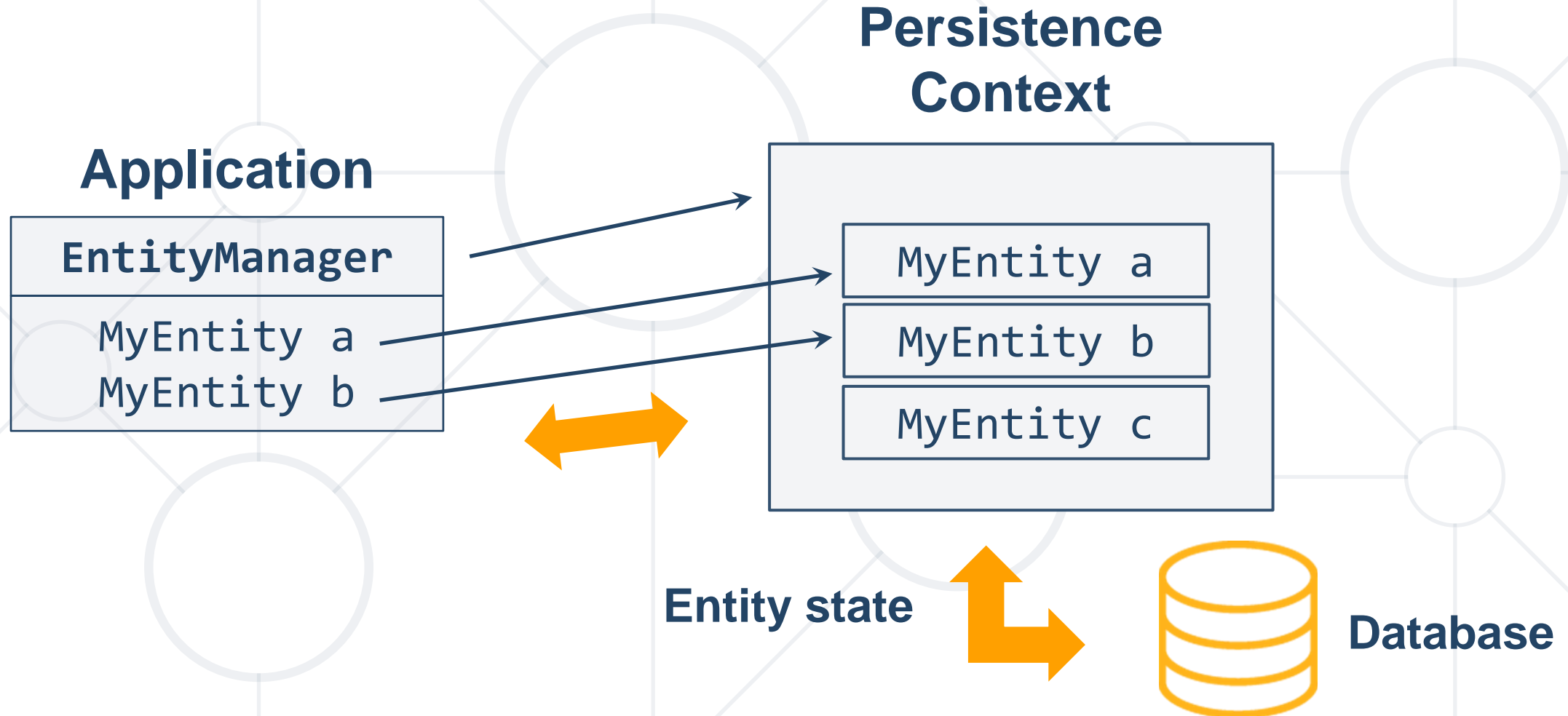```

# JPA Save Objects

| Main.java |
|---|

```java
public static void main(String[] args) {
        EntityManagerFactory emf =
Persistence.createEntityManagerFactory("school");
        EntityManager em = emf.createEntityManager();
        em.getTransaction().begin();
        Student student = new Student("Teo");
        em.persist(student);
        em.getTransaction().commit();
    }
}
```

JPA – Java Persistence API

Persistence

createEntityManagerFactory()

EntityManagerFactory

createEntityManager()

EntityManager

getTransaction()
find() / createQuery()
persist()
remove()

import
javax.persistence.*;

== **JPA**

Query

setParameter()
getResultList()
getSingleResult()
executeUpdate()

Entity

id
field1
field2

EntityTransaction

begin()
commit()
rollback()

Persistence Context (PC) and Entities

Persistence Context

Application

EntityManager
MyEntity a
MyEntity b

MyEntity a
MyEntity b
MyEntity c

Entity state

Database

Entity Object Life Cycle

new → New / Transient

persist → Managed

retrieve

merge

detach

remove → Removed

Detached

Managed → Database

clear, close

commit flush

Database
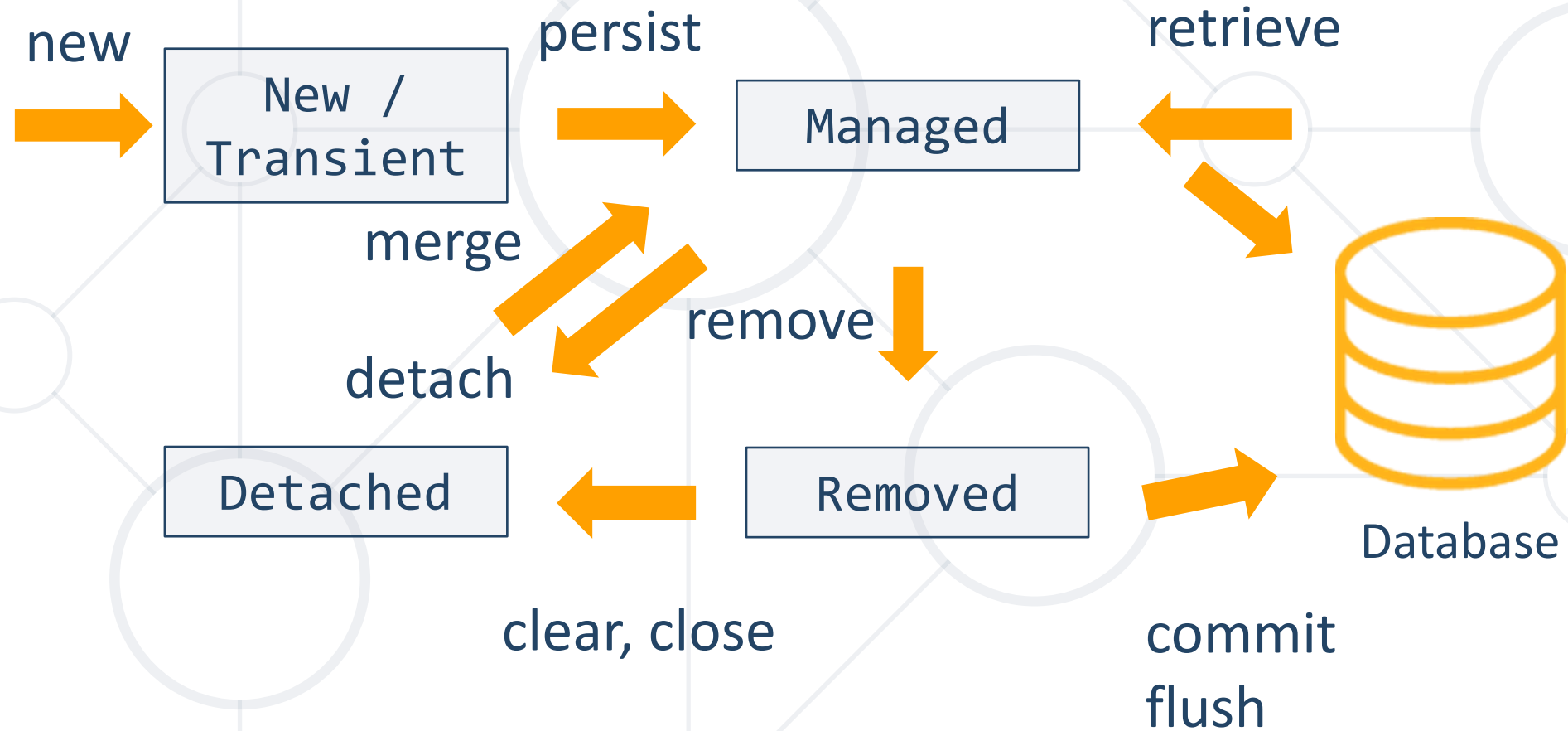
JPA Write Data Methods

- **persist()** - persists given entity object into the DB (SQL INSERT)

- **remove()** - deletes given entity into the DB (SQL DELETE by primary key)

- **refresh()** - reloads given entity from the DB (SQL SELECT by primary key)

JPA Write Data Methods

- **detach()** - removes the object from the persistence context(PC)

- **merge()** - synchronize the state of detached entity with the PC

- **contains()** - determine if given entity is managed by the PC

- **flush()** - writes the changes from PC in the database

- **find()** - execute a simple Select query by primary key

Main.java

```java
public static void main(String[] args) {
        EntityManagerFactory emf =
Persistence.createEntityManagerFactory("school");

        EntityManager em = emf.createEntityManager();
        em.getTransaction().begin();
        em.find(Student.class,1)          Get object
        em.getTransaction().commit();
    }
```

JPA Delete Objects

Main.java

```java
public static void main(String[] args) {
        EntityManagerFactory emf =
Persistence.createEntityManagerFactory("school");
        EntityManager em = emf.createEntityManager();
        em.getTransaction().begin();
        Student student = em.find(Student.class,1);
        em.remove(student);     Remove object
        em.getTransaction().commit();
    }
}
```

JPA Merge Objects

- Merges the state of **detached** entity into a **managed copy** of the detached entity.

  - Returned entity has a different Java identity than the detached one

```java
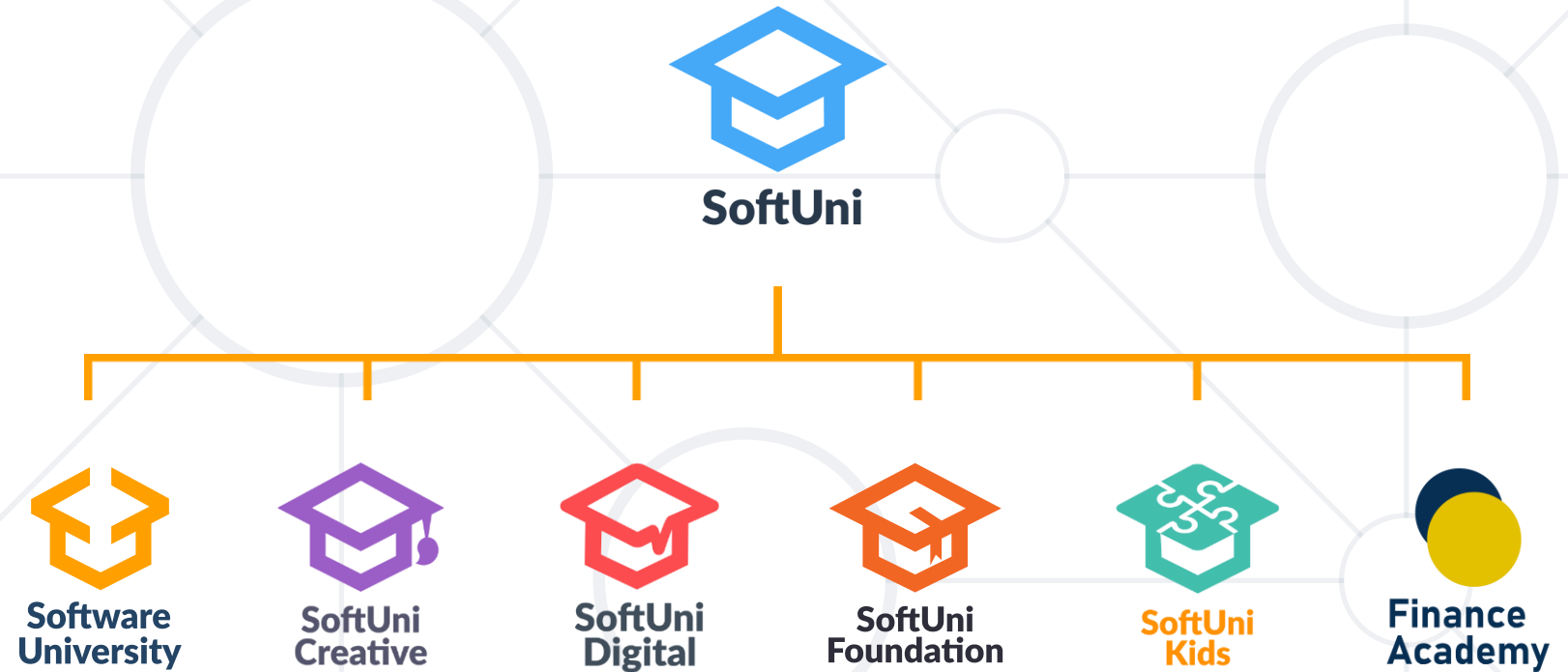public Student storeUpdatedStudent(Student student) {
    return entityManager.merge(student);
}
```

- May invoke SQL SELECT

Summary

- **Maven helps us build our project easily**
  - **Easy dependency import by XMLs**
- **Java Persistence API (JPA) is an official standard for Java ORMs**
- **Hibernate is a widely used Java ORM**
  - **Implements JPA**

Questions?

SoftUni Diamond Partners

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers

  - softuni.bg, about.softuni.bg

- Software University Foundation

  - softuni.foundation

- Software University @ Facebook

  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**

- Unauthorized copy, reproduction or use is illegal

- © SoftUni – https://about.softuni.bg/

- © Software University – https://softuni.bg