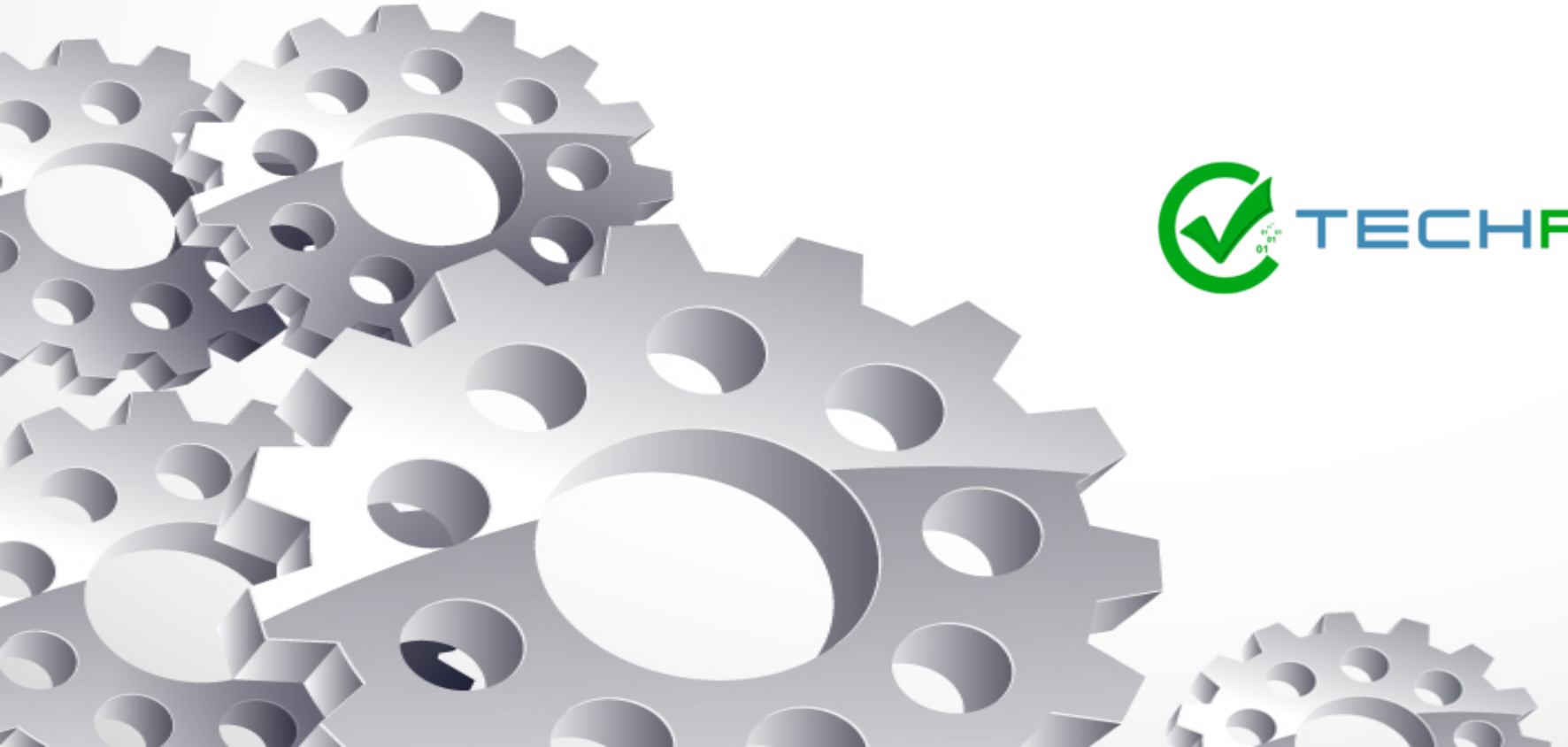
HIBERNATE

(BEKIR)



TECHPROEDUCATION

CONTENTS IN HIBERNATE COURSE

- Basic Annotations (Entity, Table, Id, Column etc.)
- Hibernate Query Language (HQL)
- Embeddable annotation
- OneToOne relation
- OneToMany, ManyToOne relation
- ManyToMany relation

CONTENTS - CONTINUE

- Fetch Types (LAZY, EAGER)
- First level ve Second Level Caches
- Details of the Get and Load method
- Hibernate Lifecycle, States: Transient, Persistent, Detached, Removed
- Hibernate Session and JPA EntityManager

General Questions

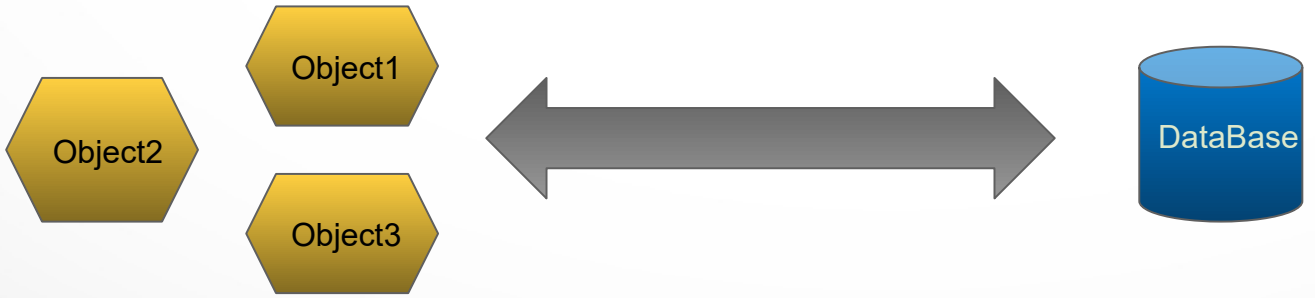
- What is Persistence?
- What is JDBC?
- What is ORM?
- What is Hibernate?
- What is JPA?

INTRO TO HIBERNATE

- OOP (Object Oriented Programming) languages (Java, C#, C++)
- Need for storing (persistence) data
  - Persistence means :you make the data permanent
- Databases and Relational Database Management System (RDBMS)
- MYSQL, PostgreSQL, SQL Server, Oracle etc.
- CRUD operations (Create, Read, Update, Delete)

INTRO TO HIBERNATE

- JDBC: Java Database Connectivity API
- ORM Object Relational Mapping
  - It is solution to handle persistent data
- ORM tools and interact with RDBMS without writing more SQL
- ORM framework / tools: Hibernate, EclipseLink, TopLink, IBatis etc.



Classes, Variables, Objects

Tables, Fields, Records (Row)

INTRO TO HIBERNATE

- JPA (Java Persistence API)
  - JPA is a specification
  - (JPA is the dance, Hibernate is the dancer.)
  - Hibernate is a super set of the JPA.
  - Hibernate has more features besides JPA has.
  - Hibernate is the most popular ORM Framework that implements JPA
  - Hibernate and JPA can’t be compared directly and they are not competitors.
  - The Java Persistence API (JPA) renamed Jakarta Persistence in 2019.

HIBERNATE JDBC RELATION

- Hibernate uses JDBC for all database communications.
- Hibernate is the layer on top of JDBC.
- Your app uses hibernate API to get and store data
- Hibernate uses JDBC API on the background for all low level JDBC work

HIBERNATE DEVELOPMENT PROCESS

- Creating maven project

- Add dependencies into pom.xml file
  - Hibernate core
  - PostgreSQL connector

- Add Hibernate Configuration File
  - Datasource
    - url, username, password , etc.

- Add Java Class and Annotate It
  - @Entity, @Table, @Id, @Column etc.

- Write Java Code to make database operations
  - Save, Fetch

MAVEN DEPENDENCIES

- Creating maven project
  - Maven dependencies

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.6.10.Final</version>
</dependency>


<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.3.6</version>
</dependency>
```

INTRO TO HIBERNATE

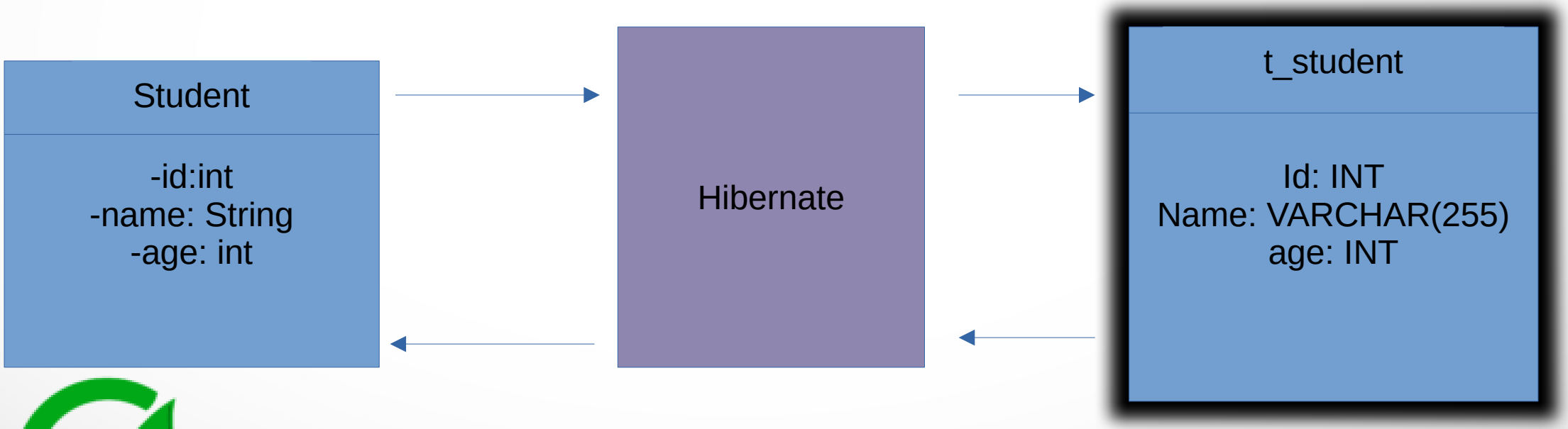
- Creating hibernate configuration file
  - hibernate.connection.driver_class:
    - org.postgresql.Driver (it changes depending on database type)
  - Database connection configuration:
    - connection.url:jdbc:postgresql://localhost:5432/dbname
    - connection.username: db_username
    - connection.password: db_password

    hibernate.dialect: (it changes depending on database type)
    - org.hibernate.dialect.PostgreSQL9Dialect
  - hbm2ddl.auto:
    - Create | update | create-drop | validate | none
  - show_sql:
    - true | false
  - format_sql:
    - true | false

TECHPROEDUCATION

Entity Class and ORM

- Entity Class: Java class that is mapped to a database table
- This is POJO (Plain Old Java Object) Class with fields and getters and setters



Mapping Types

There are 2 Options for Mapping:

- XML config file (legacy)
- Annotations (modern, used)

Annotations

- Step 1: Map class to database table
- Step 2: Fields to database columns

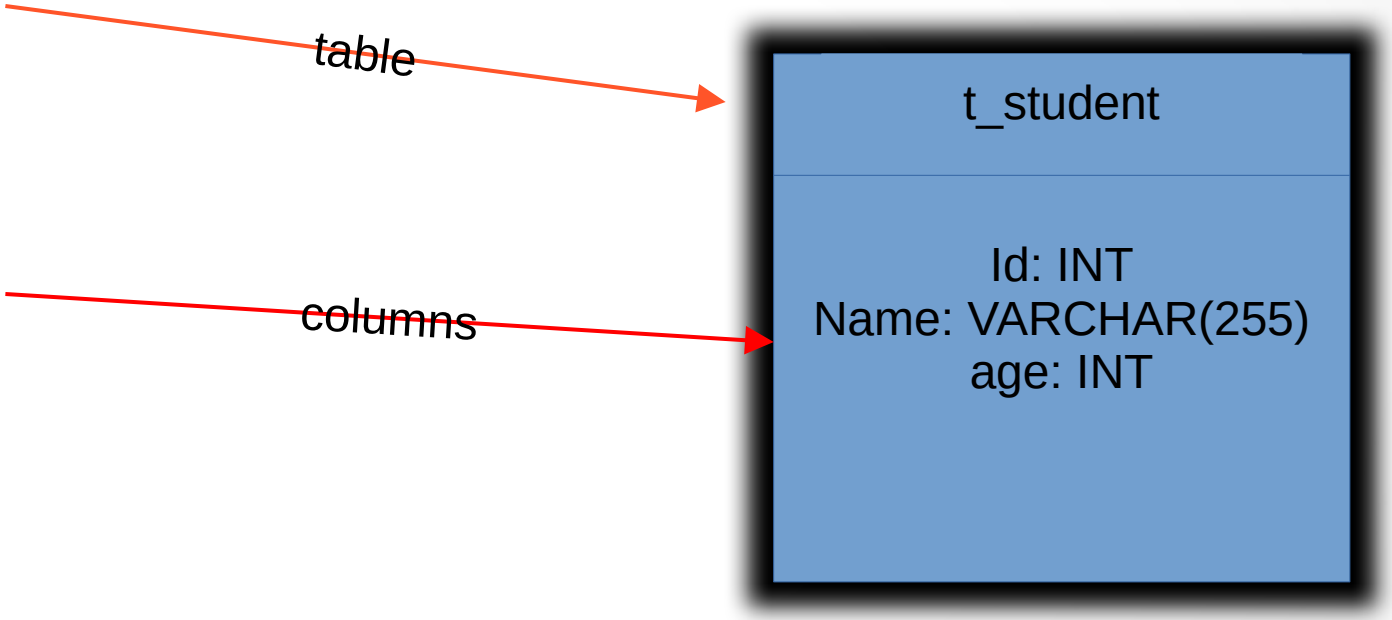
Map Class to database table
Map Fields to database columns

```java
@Entity
@Table(name="t_student")
public class Student {

@Id
private int id;

private String name;

private Integer age;
}
```



TECHPROEDUCATION

Two Key Important Structure

- SessionFactory
  - Reads hibernate configuration file
  - Creates session object
  - Only create one in your application

- Session
  - Used to get a connection with a database
  - retrieved from SessionFactory
  - Short-live object
  - It is used to save/get objects

Creating SessionFactory and Session-Codes

```
Configuration con = new
Configuration().configure("hibernate.cfg.xml").ad
dAnnotatedClass(Employee.class);

SessionFactory sf = con.buildSessionFactory();
```

Complete Code-Saving object

```
 Employee employee1 = new Employee();
employee1.setId(1);
  employee1.setName("John Coffee");
  employee1.setAge(34);

Configuration con = new
  Configuration().configure("hibernate.cfg.xml").addAnnotatedClass(Employee.class);

SessionFactory sf = con.buildSessionFactory();

Session session = sf.openSession();

Transaction tx = session.beginTransaction();

session.save(employee1);

tx.commit();

session.close();

sf.close();
```

TECHPROEDUCATION

Complete Code-Reading object

```
Employee employee= new Employee();

Configuration con = new
Configuration().configure("hibernate.cfg.xml").addAnnotatedClass(Employee.class);

SessionFactory sf = con.buildSessionFactory();

Session session = sf.openSession();

Transaction tx = session.beginTransaction();

employee= session.get(Employee.class, 1);

System.out.println(employee);

tx.commit();

session.close();

sf.close();
```

TECHPROEDUCATION

Hibernate Query Language (HQL)

- HQL is an object-oriented query language, similar to SQL.
- It uses objects and their properties Instead of table and columns.
- HQL queries are translated by Hibernate into SQL queries.
- HQL is case-insensitive except for java class and variable names

```
String hql = "FROM Employee";
Query query = session.createQuery(hql);
List<Employee> results = query.list();
```