# ОСНОВИ HIBERNATE

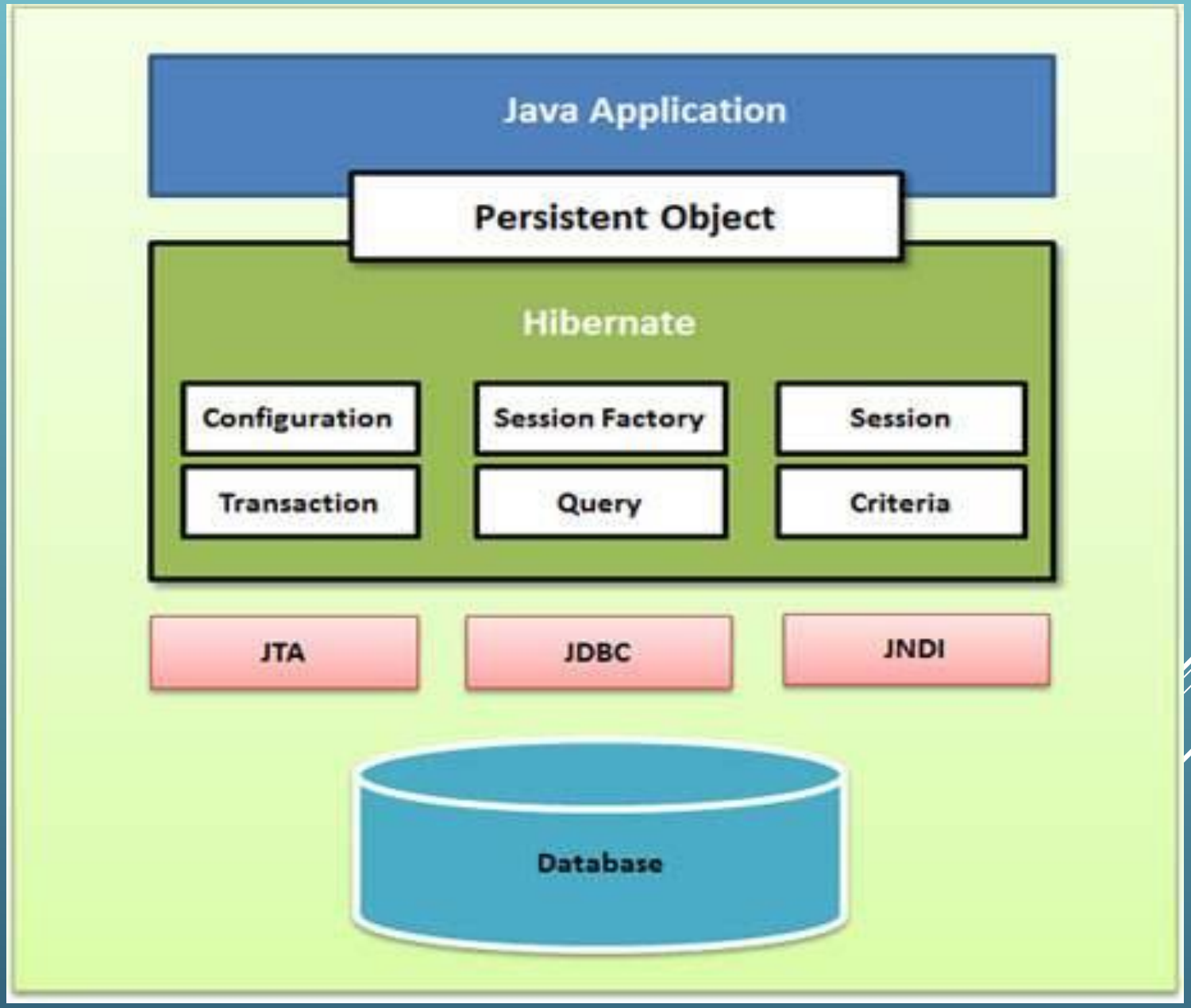
# JAVA PERSISTENCE

JPA (Java Persistence API) це специфікація Java EE і Java SE, що описує систему управління збереженням java об'єктів в таблиці реляційних баз даних в зручному вигляді. Сама Java не містить реалізації JPA, проте існує багато реалізацій даної специфікації від різних компаній (відкритих і комерційних). Це не єдиний спосіб збереження java об'єктів в бази даних (ORM систем), але один з найпопулярніших в Java і не тільки.

Hibernate одна з найпопулярніших відкритих реалізацій останньої версії специфікації (JPA 2.1). Навіть швидше найпопулярніша, майже стандарт де-факто. Тобто JPA тільки описує правила і API, а Hibernate реалізує ці описи, втім у Hibernate (як і у багатьох інших реалізацій JPA) є додаткові можливості, не описані в JPA (і не переносяться на інші реалізації JPA).

# ФРЕЙМВОРК HIBERNATE

# АРХІТЕКТУРА HIBERNATE



# CONFIGURATION OBJECT

Об'єкт конфігурації є першим в будь-якому додатку Hibernate і зазвичай створюються тільки один раз під час ініціалізації програми. Він реалізується у файлі конфігурації або властивостей і забезпечує дві основні компоненти:

- Підключення до бази даних за допомогою одного або декількох конфігураційних файлів, які підтримуються Hibernate. Ці файли hibernate.properties і hibernate.cfg.xml.
- Налаштування Mapping Class: Цей компонент створює зв'язок між класами Java і таблицями бази даних

SessionFactory є потоковим об'єкт і використовується для всіх потоків додатку. SessionFactory суперважкий об'єкт, як правило, він створюється під час запуску програми і зберігаються для подальшого використання. Якщо ви використовуєте кілька баз даних, то ви повинні створити кілька об'єктів SessionFactory.

# SESSIONFACTORY

Сесія використовується, щоб отримати фізичне з'єднання з базою даних. Об'єкт Session легкий і розроблений для того, кожен раз, коли необхідна взаємодія з базою даних. Постійні об'єкти зберігаються і витягуються за допомогою об'єкта Session. Об’єкти сесії не повинні залишатися відкритими протягом тривалого часу, вони повинні бути створені і знищені їх у міру необхідності.

# SESSION OBJECT

```
public class HibernateUtil {
    private static final SessionFactory
                sessionFactory = buildSessionFactory();

    private static SessionFactory buildSessionFactory() {    ....    }
    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
    public static void shutdown() {
        getSessionFactory().close();
    }
}
```

# HIBERNATE UTIL CLASS

```
private static SessionFactory buildSessionFactory() {
        try {
            Configuration config = new Configuration()
                    .configure(new File(HibernateUtil.class
                            .getClassLoader()
                            .getResource("hibernanate-config.xml")
                            .getFile()));
            return config.buildSessionFactory();
        }catch (Throwable ex) {
    System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }
```

# HIBERNATE UTIL CLASS

**Transaction Object:** є одиницею роботи з базою даних і більшість СУБД підтримує функціональність транзакцій. Операції в Hibernate обробляється менеджером транзакцій і транзакціями (від JDBC або JTA). Це необов'язковий об'єкт і Hibernate додаток може не використовувати цей інтерфейс, замість управління транзакціями в своєму власному коді програми.

**Query Object:** Об'єкти запиту за допомогою SQL або Hibernate Query Language (HQL) рядки для отримання даних з бази даних і створення об'єктів. Примірник запиту використовується для зв'язування параметрів запиту, обмежити кількість результатів, що повертаються запитом, і, нарешті, для виконання запиту.

**Criteria Object:** Criteria object використовуються для створення і виконання об'єктно-орієнтованих запитів для отримання об'єктів.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration SYSTEM "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.dialect">
org.hibernate.dialect.MySQLDialect</property>
        <property name="hibernate.connection.driver_class">
com.mysql.jdbc.Driver</property>        <property name="hibernate.connection.url">
jdbc:mysql://starbasi.mysql.ukraine.com.ua/starbasi_orders</property>
<property name="show_sql">true</property>
            <property name="hbm2ddl.auto">update</property>
        <property name="hibernate.connection.username">
starbasi_orders</property>
        <property name="hibernate.connection.password">
```

# КОНФІГУРАЦІЯ

```
        <property name="hibernate.connection.provider_class">
org.hibernate.connection.C3P0ConnectionProvider</property>
        <property name="hibernate.c3p0.min_size">7</property>
        <property name="hibernate.c3p0.max_size">53</property>
        <property name="hibernate.c3p0.timeout">100</property>
        <property name="hibernate.c3p0.max_statements">50</property>
        <property name="hibernate.c3p0.idle_test_period">1000</property>
        <property name="hibernate.c3p0.validate">true</property>
        <property name="hibernate.connection.provider_class">
org.hibernate.service.jdbc.connections.internal.C3P0ConnectionProvider
</property>
        <mapping resource="Student.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

# КОНФІГУРАЦІЯ

```
<?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="net.starbasic.hiberstep1.Student" table="Students">
        <meta attribute="class-description">
              This class contains the student data.
        </meta>
 <id name="id" type="int" column="ID"><generator class="native"/> </id>
        <property name="name" column="NAME" type="string"/>
            <property name="age" column="AGE" type="int"/>
            <property name="code" column="CODE" type="int"/>
    </class>
</hibernate-mapping>
```

# XML-МАПІНГ

# XML-МАПІНГ

Списки і колекції List, Set, Map і т. д.

```
<list name="certificates" cascade="all">
        <key column="employee_id"/>
        <list-index column="idx"/>
        <one-to-many class="Certificate"/>
     </list>
```

Реляційні звязки

```
<class name="Employee" table="EMPLOYEE">
.....      <many-to-one name="address" column="address"
      class="Address" not-null="true"/>
  </class>

  <class name="Address" table="ADDRESS">
    .........      <property name="zipcode" column="zipcode" type="string"/>
  </class>
```

**@Entity @Table@Id @GeneratedValue @Column:**

```
@Entity
@Table(name = "Students")
public class Student {
   @Id @GeneratedValue
   @Column(name = "id")
   private int id;
   @Column(name = "name")
   private String firstName;
```

# АНОТАЦІЇ

# HQL ПРИКЛАДИ

1)Список об'єктів з таблиці

```
from Actor
```

2)Список об'єктів зі зв'язаних таблиці

```
from Address as adr
    inner join adr.city
```

3) Умови where.

```
from Address as adr
    inner join adr.city
where adr.district='Galicia'
```

```
 from Trip as trip
     left join trip.person as person
where person.jobtitle = 'CEO'

from Trip as trip
     left join trip.person as person
where person.name = 'Able, Tony '
```

# HQL ПРИКЛАДИ