Softra Services

Learn Today, Lead Tomorrow

Hibernate Framework

Introduction to JPA

Softra Services
Learn Today, Lead Tomorrow

Module Objectives

At the end of this module, you will be able to:

- Explain the features of JPA
- Demonstrate the understanding of the JPA Architecture and the components of the architecture
- Explain the significance of the EntityManagerFactory, EntityManager and Entity
- Explain persistence unit
- Demonstrate the use of JPA

Topic Agenda

JPA Introduction

JPA Architecture & Configuration

JPA Annotations

Softra Services

Learn Today, Lead Tomorrow

JPA Introduction

Java Persistence API

- JPA is one of the specification of J2EE
- It allows the programmer to develop the persistence layer for their apps
- JPA is developed to standardize the Java ORM technologies
- JPA is not a product and can’t be used as it is for persistence
- It needs an ORM implementation to work and persist Java objects
- The major implementations of JPA specification are
  - Toplink
  - OpenJPA
  - Hibernate
  - iBATIS

JPA Contribution Areas

- The Java Persistence API
- The Query Language
- Object relational mapping metadata

Why JPA

- JPA is a standardized specification (version 1.0) and part of EJB3 specification
- Many free ORM frameworks are available which can be used to develop applications of any size
- Application developed in JPA is portable across many servers and persistence products (ORM frameworks).
- Can be used with both JEE and JSE applications
- JSE 5 features such as annotations can be used
- Provides both annotations and xml based configuration support

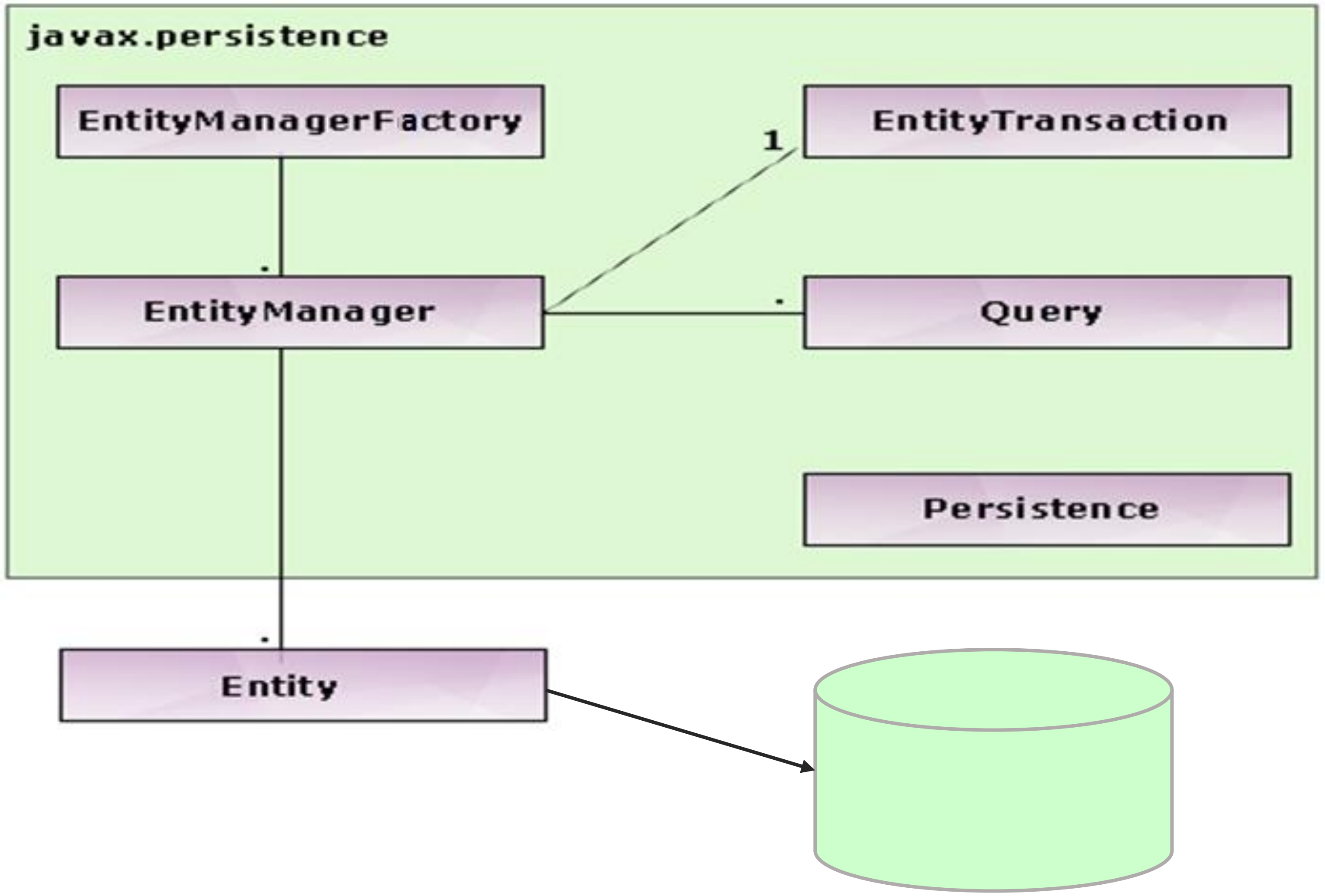
Features Of JPA

- Standardized O/R mapping
- Facilitates POJO based persistence.
- Application Server is optional with JPA.
- Support for Unidirectional and Bi-Directional relationships
- User friendly retrieval methods

JPA Architecture & Configuration

Softra Services
Learn Today, Lead Tomorrow

JPA Architecture



JPA Classes and Interfaces

- EntityManagerFactory
  - It creates and manages multiple EntityManager instances
  - EntityManagerFactory is a singleton object and represents the details of the data source
  - One factory for every data source
- EntityManager
  - It manages the persistence operations on objects
- Entity
  - Entities are the persistence objects, stores as records in the database

JPA Classes and Interfaces Continued…

- EntityTransaction
  - For each EntityManager, operations are maintained by EntityTransaction
- Persistence
  - Contain static methods to obtain EntityManagerFactory instance
- Query
  - Implemented by each JPA vendor to obtain relational objects that meet the criteria
- The above classes and interfaces are used for storing entities into a database as a record

Entity

- Entity is a lightweight persistence domain object.
- Entity class represents a database table.
- Entity instance represents a row in the database table.
- Entity uses annotations to map java fields to the database.
- Entity must follow the java bean naming conventions.
- Must implement serializable if transferred through the network

Entity Example

```
@Entity @Table(name=“Employee")
public class Employee {
    @Id @Column(name="EmpId")
     private int empId;
    @Column(name="FirstName")
     private String firstName;
    @Column(name=“LastName")
    private String lastName;
    @Column(name="Email")
    private String email;

public int getEmpId() {return empId; }
public void setEmpId(int empId) {
this.empId = empId; }
public String getFirstName() {return
firstName; }
public void setFirstName(String firstName) {
this.firstName = firstName; }
public String getLastName() { return
lastName; }
public void setLastName(String lastName) {
this.lastName = lastName; }
public String getEmail() { return email; }
public void setEmail(String email) {
this.email = email; }
}
```

Managing Entities

- Entities are managed by EntityManager
- Entity ManagerFactory is used to create EntityManager
- EntityManager is instantiated for every connection/transaction.
- EntityManager defines the methods to manage the entity life cycle
- Example:
  - persist(), remove(), find() etc

Persistence Context

- Every entity manager is associated with a persistence context
- Persistence context defines the scope under which entity instances are created, persisted and removed
- Persistence context is analogous to a container of entity instances where each entity has a live connection with the database
- Any changes done to the entity within the persistence context is reflected in the database
- There will be one and only one instance of an entity class within a persistence context
- Like a transaction persistence context can be propagated to several methods within a transaction

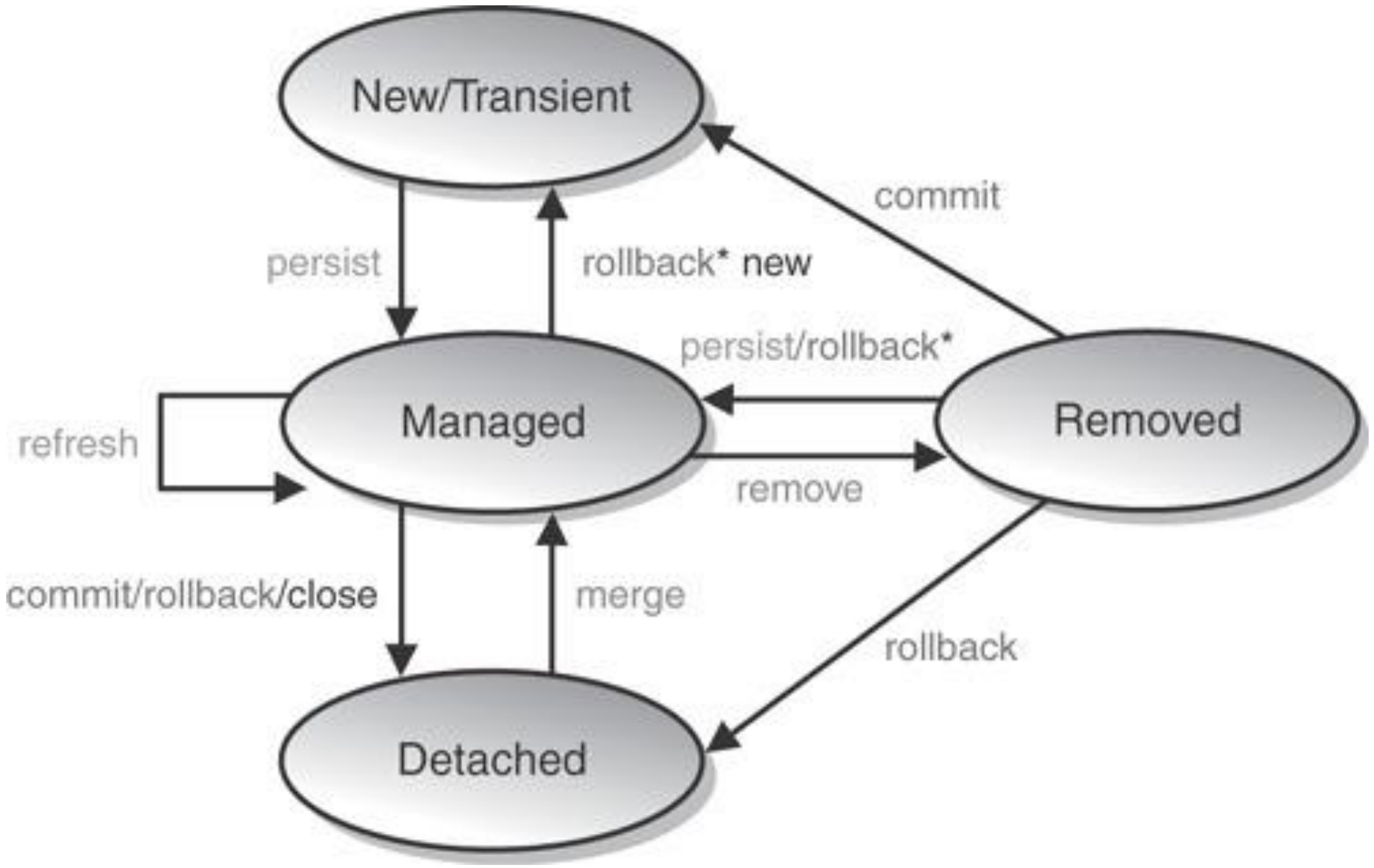
Persistence Unit

- A persistence unit defines a set of all entity classes that are managed by EntityManager instances in an application
- This set of entity classes represents the data contained within a single data store
- Persistence units are defined by the persistence.xml configuration file
- A Persistence Unit defines which classes are Persistent and which Persistence Provider should be used

Entity Lifecycle

- New/Transient
  - No persistent identity
  - Not yet associated with the persistence context
- Managed
  - Have a persistent identity and associated with the persistent context
- Detached
  - Have a persistent identity but not associated with the persistent context
- Removed
  - Have a persistent identity and also associated with the persistent context but are scheduled for removal from the database

Entity Lifecycle Continued…



Softra Services
Learn Today, Lead Tomorrow

JPA Annotations

Entity And Schema

- @Entity
  - By default, JPA assumes that a Java class is non-persistent and not eligible for JPA services unless it is decorated with this annotation
  - Use this annotation to designate a plain old Java object (POJO) class as an entity so that you can use it with JPA services.
- @Table
- @Column
  - By default, JPA assumes that an entity's name corresponds to a database table of the same name
  - And that an entity's data member names correspond to database columns with the same names

Identity And Constraints

- JPA assumes that each entity must have at least one field or property that serves as a primary key
  - Use these annotations to specify one of the following:
  - one @Id
  - multiple @Id and an @IdClass
  - one @EmbeddedId
- @GeneratedValue
  - Use the @GeneratedValue annotation if you want JPA to provide and manage entity identifiers
- @UniqueConstraint
  - Use this annotation to specify unique constraint

Softra Services
Learn Today, Lead Tomorrow

Composition

- Some objects cannot exist on their own, but can only be embedded within owning entities
- Use these annotations to specify objects that are embedded and to override how they are mapped in the owning entity's table
  - @Embeddable
  - @Embedded

@GeneratedValue: Strategy

- IDENTITY: to use a database identity column
- SEQUENCE: to use a database sequence
- TABLE: to assign primary keys for the entity using an underlying database table to ensure uniqueness
- AUTO: to choose a primary key generator that is most appropriate for the underlying database

- By default, JPA chooses the type is AUTO

@GeneratedValue: Generator

- Default: JPA assigns a name to the primary key generator it selects
- If this name is awkward, a reserved word, incompatible with a pre-existing data model, or invalid as a primary key generator name in your database, set generator to the String generator name you want to use

Softra Services
Learn Today, Lead Tomorrow

Summary

At the end of this module, you learned to:

- Explain the features of JPA
- Demonstrate the understanding of the JPA Architecture and the components of the architecture
- Explain the significance of the EntityManagerFactory, EntityManager and Entity
- Explain persistence unit
- Demonstrate the use of JPA

Softra Services

Learn Today, Lead Tomorrow

Thank You