CS489:
Applied Software Development

Prof. O. Kalu, Comp Sci, MIU © 2024

Lesson 6
Data Persistence

Wholeness

- In this lesson, we will study the concepts, principles and mechanisms for implementing Data Persistence.
- Data Persistence is an important requirement for enterprise data management and software development.
- Science of Consciousness: *Rest and activity are the natural steps of progress.*

1.1 Overview

Introduction to Data Persistence:

- Data Persistence refers to the mechanisms for storing data into non-volatile storage so that it can outlive the lifespan of a single run of the software that generated the data.
- Having Persistent Data ensures that the data remains accessible, reliable and intact even beyond the execution of the system that created it.

1.2 Types of Data Persistence

- Unstructured File-based Persistence refers to the mechanisms for writing/saving data to flat files e.g. CSV, ASCII TEXT etc.
- Semi-structured File-based Persistence refers to the mechanisms for writing/saving data in semi-structured file formats such as XML, JSON etc.
- In-memory data store – useful for caching
- **Database Persistence** – Relational, SQL databases or NoSQL databases

1.3 Data Persistence for Java Applications

- JDBC – Java Database Connectivity API
  - Java SE module: java.sql
  - Commonly used JDBC API classes and interfaces:
    - DriverManager
    - Connection
    - Statement (PreparedStatement or CallableStatement)
    - ResultSet
    - SQLException
  - See JDBC code example (in demo code pack)

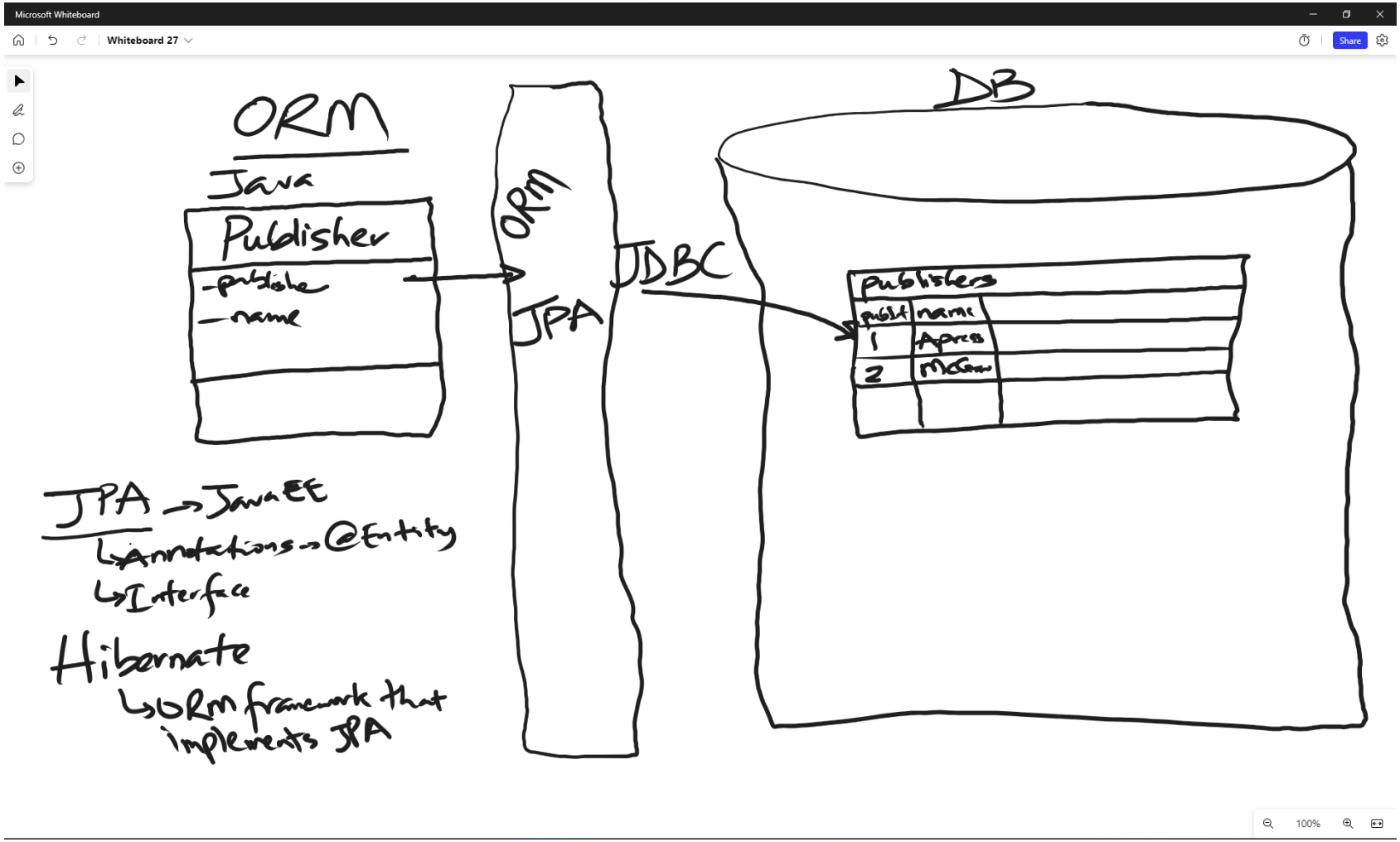
1.4 Drawbacks of using JDBC

- Impedance mismatch
- Repetitive boiler-plate code
- No compile-time error-checking of embedded SQL code

1.4 Object Relational Mapping

- Object-Relational Mapping (ORM) is a technique for translating/converting data between relational database and object-oriented programming language
- An ORM essentially creates/implements a “virtual” object database layer that can be used from within the OO programming language to access the data

1.4 Object Relational Mapping



1.5 Data Persistence for Enterprise Java Apps

- Jakarta Persistence (formerly JPA)
  - Jakarta Persistence is the specification that defines ORM for Enterprise Java applications
  - The specification document: https://jakarta.ee/specifications/persistence/3.1/jakarta-persistence-spec-3.1
  - The API documentation classes, interfaces, annotations etc:
    - https://jakarta.ee/specifications/persistence/3.1/apidocs/jakarta.persistence/module-summary.html
  - See JDBC code example (in demo code pack)

1.5 Data Persistence for Enterprise Java Apps

- Hibernate (https://hibernate.org/)
  - Hibernate is an open-source Object-Relational Mapping (ORM) framework for implementing data persistence for enterprise application for the Java platform
  - Hibernate is an implementation of the Jakarta Persistence specification – it is a JPA provider
  - Other alternatives are: EclipseLink, JOOQ, Ebean, Apache OpenJPA etc.

1.6 Spring Platform, Spring Data and Spring Data JPA

- Spring platform (https://spring.io/)
  - Provides tools, frameworks and libraries for building enterprise-grade applications for the JVM
- Spring Data
  - Spring module (project) useful for implementing data access to a variety of underlying data stores.
- Spring Data JPA
  - Part of the Spring Data family; for implementing JPA-based data access components for Spring applications
  - Uses Hibernate as the default JPA provider

1.8 Demo Implementation of Data Persistence

- In this demo coding, we implement basic data persistence for the City Library app using Spring Boot, Spring Data JPA and performing the CRUD operations on a single entity class named, Publisher

1.8 Demo Implementation of Data Persistence

- Use Spring Initializr to create a CLI application
- Print “Hello. Welcome to Data Persistence Demo”

1.8 Demo Implementation of Data Persistence

- Add Spring Boot Starter dependencies:
  - spring-boot-starter-data-jpa
  - MySQL Connector/J (JDBC Driver for MySQL Database)

1.8 Demo Implementation of Data Persistence

- Data Validation:
  - Database Data validation
    - Column constraints
  - Java Bean Validation (JSR303 validators)
    - @NotNull
    - @NotEmpty
    - @NotBlank
    - @Size, @Min, @Max etc.
    - @DateTimeFormat

1.8 Demo Implementation of Data Persistence

- Mapping Entity Relationships:
  - One-to-one association
    - Unidirectional
    - Bidirectional (Note: mappedBy)
  - One-to-many, Many-to-one association
    - Unidirectional
    - Bidirectional (Note: mappedBy)
  - Many-to-many association
    - Unidirectional, Bidirectional
    - Note: Requires JoinTable

1.8 Demo Implementation of Data Persistence

- Applying Cascade operations:
  - Cascade Types
    - CascadeType.PERSIST
    - CascadeType.MERGE
    - CascadeType.ALL
- Data Fetch strategies
  - Eager fetching versus Lazy fetching
  - Fetch Types
    - FetchType.EAGER
    - FetchType.LAZY
  - Defaults – x-to-one (FetchType.EAGER), x-to-many (FetchType.LAZY)

1.8 Demo Implementation of Data Persistence

- Mapping Entity Inheritance Relationships:
  - Single-Table per hierarchy

```java
@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
public class MyProduct {
    @Id
    private long productId;
    private String name;

    // constructor, getters, setters
}
```

```java
@Entity
public class Book extends MyProduct {
    private String author;
}
```

```java
@Entity
public class Pen extends MyProduct {
    private String color;
}
```

- Can specify a Discriminator column (see code example)

1.8 Demo Implementation of Data Persistence

- Querying Data:
  - JPQL Queries

```
@Query("select p from Publisher where p.id = :id")
```

  - Native (SQL) Queries

```
@Query("select * from publishers p where p.id = :id", nativeQuery = true)
```

  - Spring Data JPA Query Methods

```
Optional<Publisher> findPublisherById(int id);
```

Main Point

- Data Persistence is an important requirement for enterprise data management and software development.
- Science of Consciousness: *Rest and activity are the natural steps of progress.*

Lab 6

- Implement Data Persistence for the ADS system
- See the Sakai Assignment – Lab6

CS489:
Applied Software Development