Java Persistence API 2.2

INTRODUCTION

Antonio Goncalves
JAVA CHAMPION

@agoncal www.antoniogoncalves.org

Course Outline

Introduction

Understanding Java Persistence API

Mapping and Managing Entities

Querying Entities

Relationships and Inheritance

Entity Lifecycle, Callbacks, and Listeners

Java Persistence API 2.2 within Java EE 8

Audience

Technical

Developer

Architect

Technical Level

Intermediate

Java

Relational Database

Mac OS X

Usage of Java Persistence API

Simple

Complex

Cloud

Relational Databases

Overview

What is persistence?

How do we usually persist data?

What’s wrong with the way we do?

How can Java Persistence API help?

What Is Persistent?

Data

Storage

Central point

Manipulating data

Relational databases

Demo

Web application

eCommerce website for books and CDs

Manipulates data from the database

Create, search, update or delete

Manipulating Persisted Data without JPA

Objects

Relational databases

Object-relational mapping

Rely on external frameworks

JDBC

Java Database Connectivity

Java-based data access technology

Methods for querying and updating data

Part of JDK 1.1 on 1997

JDBC 4.3 since Java 9

Robust

Low level and verbose

A Book Class

```
public class Book {

  private Long id;
  private String title;
  private String description;
  private Float unitCost;
  private String isbn;

  // Constructors, getters & setters
}
```

A Main Class Manipulating a Book

```
public class Main {

  public static void main(String[] args) {

    persistBook(new Book(1L, "H2G2", "Best Scifi Book",
                         12.5f, "1234-5678-5678", 247));

    Book book = findBook(1L);

    System.out.println(book);
  }
```

Getting a Database Connection

```
static {
  try {
    Class.forName("org.h2.Driver");
  } catch (ClassNotFoundException e) {
    e.printStackTrace();
  }
}

private static Connection getConnection()
                                throws SQLException {
  return DriverManager.getConnection(
            "jdbc:h2:mem:module01-db");
}
```

Persisting a Book to the Database

```
private static void persistBook(Book book) {

  String query = "INSERT INTO BOOK (ID, TITLE, DESCRIPTION,
                  UNITCOST, ISBN)  VALUES (?, ?, ?, ?, ?)";

  try (PreparedStatement stmt = getConnection().prepareStatement(query)){

    stmt.setLong(1, book.getId());
    stmt.setString(2, book.getTitle());
    stmt.setString(3, book.getDescription());
    stmt.setFloat(4, book.getUnitCost());
    stmt.setString(5, book.getIsbn());

    stmt.executeUpdate();
  }
}
```

Retrieving a Book from the Database

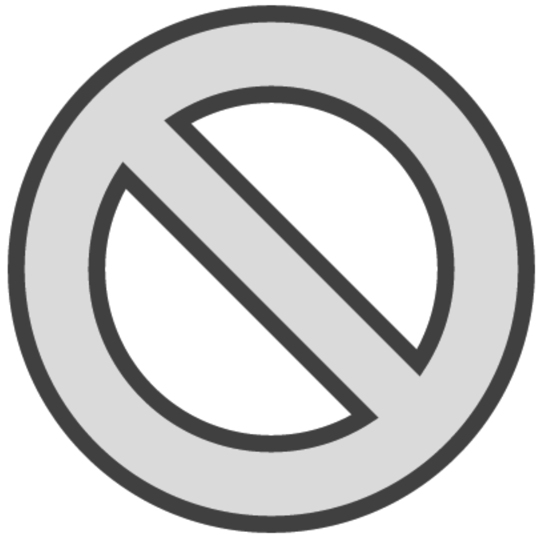
```
private static Book findBook(Long id) {
  Book book = new Book(id);
  String query = "SELECT ID, TITLE, DESCRIPTION,
                 UNITCOST, ISBN FROM BOOK WHERE ID = ?";

  try (PreparedStatement stmt = getConnection().prepareStatement(query)){

    stmt.setLong(1, id);
    ResultSet rs = stmt.executeQuery();

    while (rs.next()) {
      book.setTitle(rs.getString("TITLE"));
      book.setDescription(rs.getString("DESCRIPTION"));
      book.setUnitCost(rs.getFloat("UNITCOST"));
      book.setIsbn(rs.getString("ISBN"));
    } }
  return book;
}
```

What’s Wrong with JDBC?



SQL is not Java

JDBC is a low level API

SQL is not easy to refactor

JDBC is verbose

Hard to read

Hard to maintain

Manipulating Persisted Data with JPA

Standard

Object-relational mapping

Meta-data mapping

Removes boiler plate code

(C)reate, (R)ead, (U)pdate, (D)elete

Object-oriented query language

A Book Entity

```
@Entity
public class Book {

  @Id
  private Long id;
  private String title;
  private String description;
  private Float unitCost;
  private String isbn;

  // Constructors, getters & setters
}
```

A Main Class Manipulating a Book Entity

```
public class Main {

  public static void main(String[] args) {

    persistBook(new Book(1L, "H2G2", "Best Scifi Book",
                         12.5f, "1234-5678-5678", 247));

    Book book = findBook(1L);

    System.out.println(book);
  }
```

A Main Class Manipulating a Book Entity

```
  private static EntityManagerFactory emf =
    Persistence.createEntityManagerFactory("module01PU");

  private static EntityManager em = emf.createEntityManager();

  private static void persistBook(Book book) {
    em.persist(book);
  }

  private static Book findBook(Long id) {
    return em.find(Book.class, id);
  }
}
```

Advantages of JPA

No manual mapping

No SQL statements

Interaction through EntityManager

Non intrusive

Lightweight

Elegant API

Powerful

Demo

Inserting and retrieving a book

JDBC

Book class

Low-level JDBC API

JPA

Annotation

High-level Entity Manager API

Summary

Data is crucial

We need to store data

We need to manipulate data

Level of abstraction

Object Relational Mapping

Simplicity

Powerful

Next Module

Object-relational mapping

JPA is, does, does not

JPA specification

Programming model

Architectural layers