Drives the Complexity out of Database Integration on the Android Platform

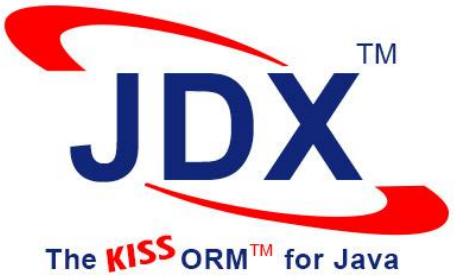
By

Damodar Periwal

Founder

1999 S Bascom Avenue, Suite 700
Campbell, CA 95008, USA
Phone: +1-408-282-3606
Email: dperiwal@softwaretree.com
URL: http://www.softwaretree.com

- Data integration software company specializing in **Object Relational Mapping (ORM)** technology
- JDX, the core ORM technology, simplifies integration of Java programs with relational databases by eliminating endless lines of SQL code
- JDX ORM is powerful, practical, and patented
- JDX ORM has been adapted for the .NET and Android Platforms



- JDX helps achieve significant reductions in overall time, risk and cost associated with Java/Database programming
- Released in 1998
- Customers include British Telecom, Xerox, Los Alamos National Labs, Electronic Arts, Darden Business School, and UAB Hospital System

- NJDX helps achieve significant reductions in overall time, risk and cost associated with .NET/Database programming
- Released in 2005
- NJDX has been tightly integrated with Visual Studio .NET and can be used with any CLR-based language including C# and VB.NET

- JDXA is a simple yet powerful, flexible, and lightweight ORM product for the Android platform
- JDXA helps achieve significant reductions in overall time, risk and cost associated with Android/SQLite programming
- Released in 2015
- Comes with many Android platform-specific utility classes to facilitate the easy and speedy development of mobile apps

Testimonials for our ORM products

- ***I'm more impressed with the power and depth of your software every day.*** - Dr. Dave Forslund, Deputy Director, Los Alamos National Laboratory

- ***I have evaluated JDXA ORM for Android and am very impressed by the product's powerful features, performance, and simplicity.*** - Surojit Pakira, a senior Android application developer

- ***JDXA is one of the easiest Android ORM frameworks I have worked with so far. I was up and running with JDXA in literally a few minutes. It's really simple to use and understand. If you are looking for a simple, yet powerful ORM framework that can significantly accelerate your Android app development process, choose JDXA.*** - Lakitha Samarasinghe, Mobile Tech Lead, Fidenz

More Testimonials

- ***JDX simplified the rapid evolution of our application design by easily facilitating the mapping and database schema changes. JDX has met our performance expectations very well*** – Greg Ball Director, Darden Information Services

- ***The reverse-engineering capabilities of JDX really set it apart from the others*** - Kevin Leitch, Java System Architect

- ***We've tried Oracle but couldn't achieve what we've achieved with JDX*** - Chee-Beng Chay, Director, PalmWindow, Singapore

- ***I did not encounter any modeling or query requirements in our complex application for which JDX did not have a solution*** - Richard Brewster, News Corporation

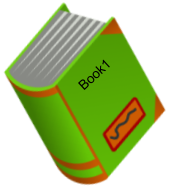
Role of ORM in Application Architecture

- **Common Application Design and Usage Pattern**
  - Object-Oriented (OO) Programs using RDBMS as persistence storage for business (domain) objects

- **Business (Domain) Object Examples**
  - *A Twitter App:* User, Tweet, ReTweet
  - *A Fitness App:* User, WhatToTrack, TrackedInfo, FitnessGoal, VitalReadingsLog
  - *A ToDo App:* User, Task, TodoList, Location, TaskDisposition
  - *A Travel Journal App:* User, Location, Attractions, Restaurants, ActivityLog

An Example Business Domain Object Model

Source: http://examples.javacodegeeks.com/android/core/database/android-database-example/




```
package com.javacodegeeks.androiddatabaseexample;

public class Book {

        private int id;
        private String title;
        private String author;

        public Book() {}

        public Book(String title, String author) {
                super();
                this.title = title;
                this.author = author;
        }

        public int getId() {
                return id;
        }
```

Android Database Access Code Without ORM

Source: http://examples.javacodegeeks.com/android/core/database/android-database-example/

```
        // SQL statement to create book table
        String CREATE_BOOK_TABLE = "CREATE TABLE books ( " + "id
INTEGER PRIMARY KEY AUTOINCREMENT, " + "title TEXT, " + "author TEXT )";
        db.execSQL(CREATE_BOOK_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // drop books table if already exists
        db.execSQL("DROP TABLE IF EXISTS books");
        this.onCreate(db);
    }

    public void createBook(Book book) {
        // get reference of the BookDB database
        SQLiteDatabase db = this.getWritableDatabase();

        // make values to be inserted
        ContentValues values = new ContentValues();
```

Role of ORM in Application Architecture …

- **The Problems**
  - Difficult to bridge the object-relational paradigm (impedance mismatch)
  - Significant pieces of complex code being repeatedly developed
  - Tedious, error-prone, and time-consuming exercise
  - High cost of development and maintenance

- **Best Practice Design Pattern**
  - Persistence framework based on ORM functionality
  - Eliminate complex, non-intuitive and error-prone JDBC/SQL code
  - Good isolation of persistence layer eliminates bigger problems

Android Database Access Code With JDXA ORM

JDXA ORM spec

```
CLASS .Book TABLE Books
  PRIMARY_KEY id
```

Database Access Code with JDXA ORM

```
public void createBook(Book book) throws Exception {
        jdxHelper.insert(book, true);
}

public Book readBook(int id) throws Exception {
        return (Book) jdxHelper.getObjectById(bookClassName, "id=" + id, false,
                        null);
}

public List getAllBooks() throws Exception {
        return jdxHelper.getObjects(bookClassName, null);
}

public void updateBook(Book book) throws Exception {
        jdxHelper.update(book, true);
```

KISS Principle

- Keep It Simple, Stupid
- Keep It Simple, Silly
- Keep It Short and Simple
- Keep It Small and Simple
- Keep It Simple and Straightforward

Most systems work best if they are kept simple rather than made complicated

What are the *KISS* Principles for ORM?

KISS Principles for ORM

#1

Solve the most important problem
(object relational impedance mismatch)
in the simplest possible way

The ORM product focuses on the most important problem and solves it efficiently

KISS Principles for ORM

#2

Don’t make the solution more complex than the original problem

Rather than becoming a development headache, the ORM improves developer productivity

KISS Principles for ORM

#3

Be completely non-intrusive to the object model

A clean object model helps in easier implementation and smoother evolution of business logic

KISS Principles for ORM

#4

Give full flexibility in object modeling

Adherence to a true domain model helps in better design and integration of the application

KISS Principles for ORM

#5

Make it easy to define, modify, comprehend, and share the mapping specification

The ORM system is easy to understand, use, and manage

KISS Principles for ORM

#6

Avoid source code generation for data access

Creates a simpler, cleaner, and more dynamic solution

KISS Principles for ORM

#7

Keep the mapping engine as much stateless as possible

The mapping engine remains simple and focused without creating unnecessary runtime overhead

KISS Principles for ORM

#8

No mind reading

The mapping engine does not cause data corruption. The user remains firmly in control. The usage of an ORM engine is simple and straightforward.

KISS Principles for ORM

#9

Avoid creating a new query language

Fast learning curve. Easy-to-understand programs. Avoiding the overhead related to query parsing and compilation speeds up internal implementation.

KISS Principles for ORM

#10

Stick to 90/90 rule about product features

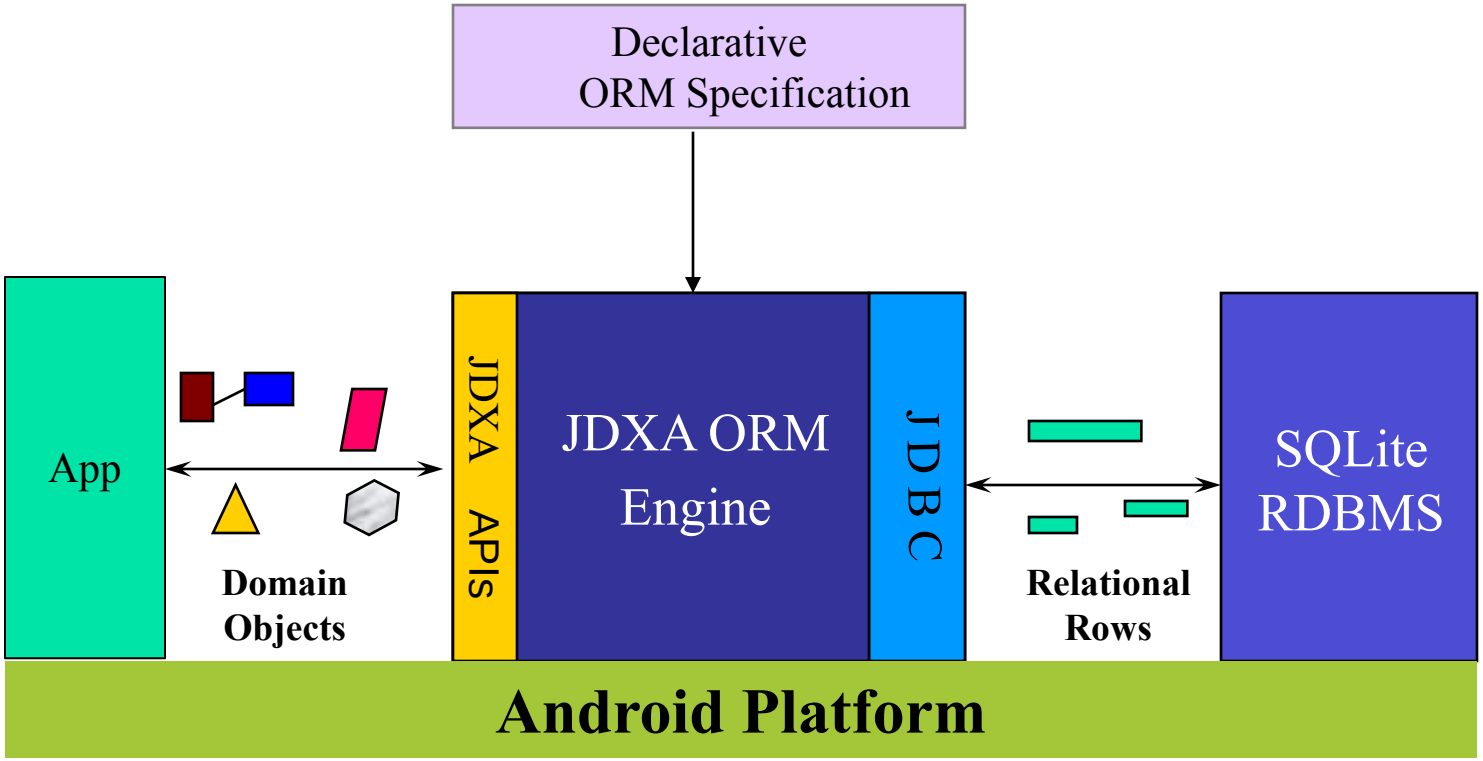
A practical product that is easy-to-understand and use. Implementation is not overloaded with unnecessary or rarely-used features.

Simplicity is the ultimate sophistication

Leonardo da Vinci

JDXA™
The KISS ORM™ for Android

ORM Architecture



There are just 3 simple steps to use JDXA ORM

1. Define domain object model (Java classes)

2. Define a declarative ORM specification textually

3. Develop apps using intuitive and powerful JDXA APIs

Features/Benefits

- Declarative mapping specification based on a simple grammar
  - Benefit: Mapping is easy to define, generate, modify & comprehend
- Provides simple, non-intrusive, and dynamic programming model
  - Benefit: Increased developer productivity
- Handles complex object structures and class-hierarchies
  - Benefit: Greater flexibility working with objects
- Powerful query facilities including object-streaming, named queries, and object caching
  - Benefit: More flexible, sophisticated, and faster apps

Features/Benefits

- Supports POJO (Plain Old Java Objects) friendly non-intrusive programming model which does not require you to change your Java classes in any way
  - Benefits:
    - No need to subclass your domain classes from any base class
    - No need to clutter your source code with annotations
    - No source code generation (No need for DAO classes)
    - No pre-processing or post-processing of your code
    - Clean architecture improves developer productivity and code maintainability

JDXA™

The KISS ORM™ for Android

Features/Benefits

Object Modeling Flexibility

Class Hierarchies

One-to-Many Relationships

BYVALUE Relationships

Implicit Attributes

One-to-One Relationships

Many-to-Many Relationships

BYREFERENCE relationships

Persistence By Reachability

JDXA™
The KISS ORM™ for Android

Features/Benefits

Query Flexibility

Shallow Query	Deep Query
Directed Query	Lazy Fetches
Named Query	Positional Query
Aggregate Query	Streaming Query
Asynchronous Query	Polymorphic Query
Query by Identity	Path Expressions

Features/Benefits

- Android specific utility classes for
  - Schema creation/population
  - ListActivity
  - Asynchronous queries
  - Streaming objects
  - Sequence generators
  - Object graph display
  - JDXHelper – a useful façade over the core ORM methods
  - Benefit: Create flexible apps quickly
- Support for persistence of JSON objects
  - Benefit: Easily create apps utilizing web services
- Extensive documentation and many working examples
  - Benefit: Easy-to-learn and easy-to-use

JDXHelper Methods

Partial List

```
public List getObjects(String className, String predicate)

public List getObjects(String className, String predicate, long
maxObjects, boolean deep, List details)

public Object getObjectById(String className, String
primaryKeyPredicate, boolean deep, List details)

public void insert(Object object, boolean deep)

public void update(Object object, boolean deep)

public void delete(Object object, boolean deep)

public void delete2(String className, String predicate)
```

Could be a list of objects

JDXHelper Methods

Partial List (Contd.)

```
public int getObjectCount(String className, String attribName,
String predicate)

public synchronized long getNextSeq(String seqName, long
increment)

public long SQLStatement(String statement, long statementFlags)
```

JDXS Methods

Partial List

```
public List query(String className, String predicate, long
maxObjects, long queryFlags, List queryDetails)

public void insert(Object object, long insertFlags, List
insertDetails)

public void update(Object object, long updateFlags, List
updateDetails)

public void delete(Object object, long deleteFlags, List
deleteDetails)

public void delete2(String className, String predicate, long
deleteFlags)
```

Could be a list of objects

JDXA™

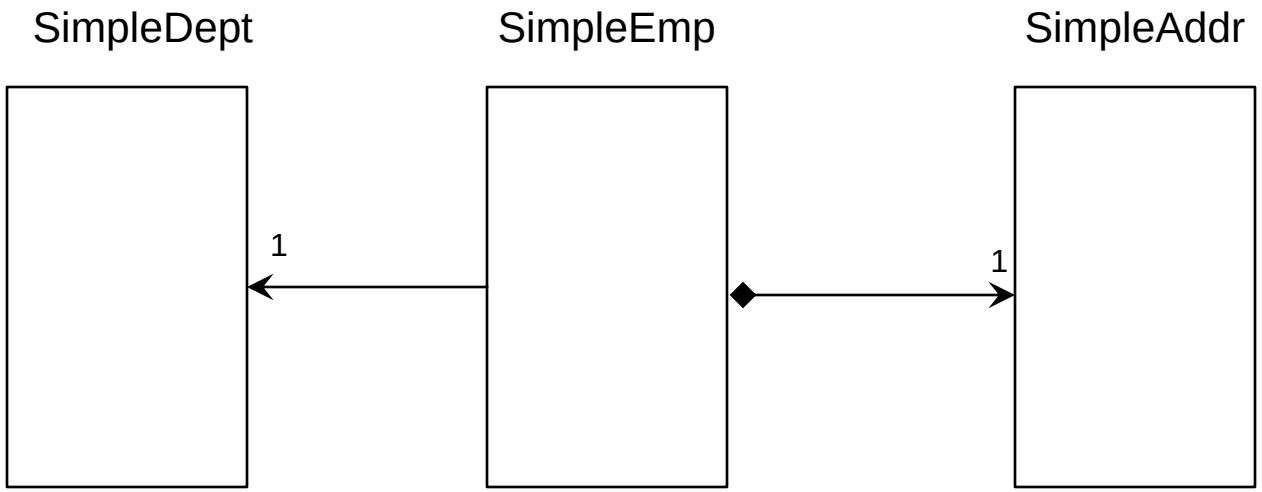
The KISS ORM™ for Android

Code Snippets

JDXA™

The KISS ORM™ for Android

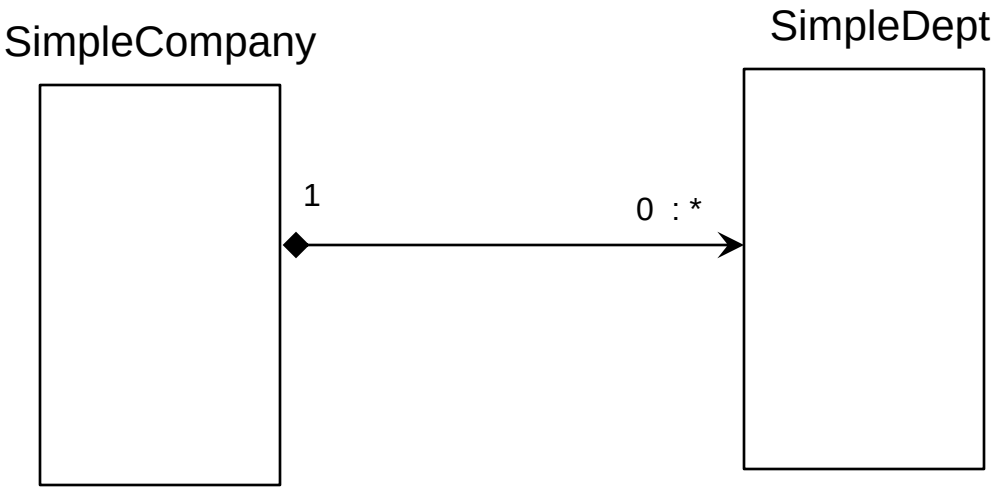
One-to-One Relationship



An employee works in a department (BYREFERENCE Relationship)

An employee has an address (BYVALUE Relationship)

One-to-Many Relationship



A company has many departments (BYVALUE Relationship)

Online Code Snippets

JDXA ORM code snippets provided

for many different object modeling and usage pattern

at the following url

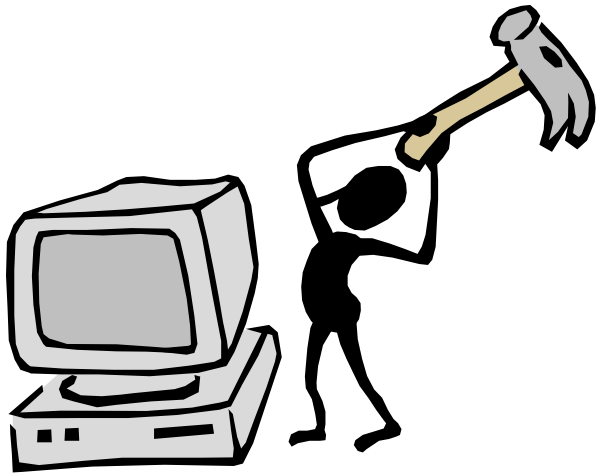
http://www.softwaretree.com/v1/products/jdxa/code-snippets.html

Technology Summary

- A simple and flexible ORM framework is a fundamental need for present and future application architectures to access SQL relational databases like SQLite
- JDXA ORM is simple, flexible, non-intrusive, and lightweight
- JDXA provides a powerful set of practical ORM features
- JDXA improves developer productivity by maximizing code reuse, maintainability, and reliability
- JDXA helps achieve significant reductions in overall time, risk, and cost associated with Android app development

Welcome JDXA™ and say goodbye to

The KISS ORM™ for Android



Mr. Code Struggle

Ms. Project Delay

Questions?

Thank You!

Resources

Software Tree Website

http://www.softwaretree.com

More Information on JDXA

Free 30-day Trial Download

White paper: KISS Principles for ORM

Software Tree™
Simplifying Data Integration

Additional Slides

JDXA Tutorials

	**JDXA, simple yet powerful ORM library for Android**
Lakitha Samarasinghe	**URL: http://wp.me/p5X7bM-2w**

KISS Principles for ORM

- Solve the most important problem (object relational impedance mismatch) in the simplest possible way
- Don’t make the solution more complex than the original problem
- Be completely non-intrusive to the object model
- Give full flexibility in object modeling
- Make it easy to define, modify, comprehend, and share ORM specification
- Avoid source code generation for data access
- Keep the mapping engine as much stateless as possible
- No mind reading
- Avoid creating a new query language
- Expose small number of simple and consistent APIs.

More ….

KISS Principles for ORM…

- Absorb database-specific dependencies in the internal implementation.
- Provide simple and intuitive pass-thru mechanisms for accessing databases directly.
- Optimize data access logic automatically.
- Stick to 90/90 rule about product features.
- Keep the internal implementation simple, extensible, and efficient.
- Offer intuitive tools to deal with object models, database schema, and the mapping.
- Provide a straightforward installer, lucid documentation, and readymade examples.

More Testimonials

- ***We could not have finished our project in time without JDX*** - Paul Quirk, PMSC, Australia
- ***JDX is a top-tier OR-Mapping technology. Simple definition of the mapping in the text form is very innovative, powerful and interesting -*** Lubos Hartman, Software Architect for J2EE, Unicorn, Czech Republic
- ***We are very impressed with JDX. In building a large-scale Java application, the object-oriented access to DBMS, eliminating the need of SQL code, is extremely important*** - Alex Elkin, VP of Engineering, IntelliFrame Corporation
- ***Personally I find JDX to be the best among all that is existing out there. I have already evaluated it on the local system and its performance is excellent -*** Niranjan Joshi, Java Consultant

More Testimonials

- ***I wish we had known about JDX before getting too deep into our own home-grown mess of the complex object-relational mapping code*** - Name withheld