Data access object

From Wikipedia, the free encyclopedia

In computer software, a **data access object** (**DAO**) is an object that provides an abstract interface to some type of database or other persistence mechanism. By mapping application calls to the persistence layer, the DAO provides some specific data operations without exposing details of the database. This isolation supports the Single responsibility principle. It separates what data access the application needs, in terms of domain-specific objects and data types (the public interface of the DAO), from how these needs can be satisfied with a specific DBMS, database schema, etc. (the implementation of the DAO).

Although this design pattern is equally applicable to the following: (1- most programming languages; 2- most types of software with persistence needs; and 3- most types of databases) it is traditionally associated with Java EE applications and with relational databases (accessed via the JDBC API because of its origin in Sun Microsystems' best practice guidelines[1] "Core J2EE Patterns" for that platform).

Contents

Advantages

The advantage of using data access objects is the relatively simple and rigorous separation between two important parts of an application that can but should not know anything of each other, and which can be expected to evolve frequently and independently. Changing business logic can rely on the same DAO interface, while changes to persistence logic do not affect DAO clients as long as the interface remains correctly implemented. All details of storage are hidden from the rest of the application (see information hiding). Thus, possible changes to the persistence mechanism can be implemented by just modifying one DAO implementation while the rest of the application isn't affected. DAOs act as an intermediary between the application and the database. They move data back and forth between objects and database records. Unit testing the code is facilitated by substituting the DAO with a test double in the test, thereby making the tests non-dependent on the persistence layer.

In the non specific context of the Java programming language, Data Access Objects as a design concept can be implemented in a number of ways. This can range from a fairly simple interface that separates the data access parts from the application logic, to frameworks and commercial products. DAO coding paradigms can require some skill. Technologies like Java Persistence API and Enterprise JavaBeans come built into application servers and can be used in applications that use a JavaEE application server. Commercial products like TopLink are available based on Object-relational mapping (ORM). Popular open source ORM products include Doctrine, Hibernate, iBATIS and JPA implementations such as Apache OpenJPA.

Disadvantages

Potential disadvantages of using DAO include leaky abstraction, code duplication, and abstraction inversion. In particular, the abstraction of the DAO as a regular Java object can hide the high cost of each database access, and can also force developers to trigger multiple database queries to retrieve information that could otherwise be returned in a single operation with normal SQL set operations. If an application requires multiple DAOs, one might find oneself repeating essentially the same create, read, update, and delete code for each DAO. This boiler-plate code may be avoided however, by implementing a generic DAO that handles these common operations.[2] Time consumption is moderate.

Tools and frameworks

- ODB compiler-based object-relational mapping (ORM) system for C++
- ORMLite Lightweight object-relational mapping (ORM) Framework in Java for JDBC and Android[3]
- Microsoft Entity Framework
- DBIx::Class (http://www.dbix-class.org/about.html) object-relational mapping (ORM) module for Perl

See also

- Create, read, update and delete (CRUD)
- Data access layer
- Service Data Objects

References

1. "Core J2EE Patterns - Data Access Objects". Sun Microsystems Inc. 2007-08-02.
2. See http://www.ibm.com/developerworks/java/library/j-genericdao/index.html for workarounds
3. Hodgson, Kyle; Reid, Darren. *ServiceStack 4 Cookbook*. Packt Publishing Ltd. p. Chapter 4. ISBN 9781783986576. Retrieved 22 June 2016.

External links

- *Java Persistence - The DAO Design Pattern* (http://tutorials.jenkov.com/java-persistence/dao-design-pattern.html)
- PHP best practices (Use Data Access Objects (DAO)) (http://www.odi.ch/prog/design/php/guide.php)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Data_access_object&oldid=769223382"

Categories: Software design patterns | Architectural pattern (computer science)