DAO Design Pattern

HUBHAM — 8 COMMENTS

DAO stands for Data Access Object. DAO Design Pattern is used to separate the data persistence logic in a separate layer. This way, the service remains completely in dark about how the low-level operations to access the database is done. This is known as the principle of **Separation of Logic**.

Patterns

- Adapter
- Composite
- Proxy
- Flyweight
- Facade

Table of Contents [hide]

DAO Design Pattern

With DAO design pattern, we have following components on which our design depends:

- The model which is transferred from one layer to the other.
- The interfaces which provides a flexible design.
- The interface implementation which is a concrete implementation of the persistence logic.

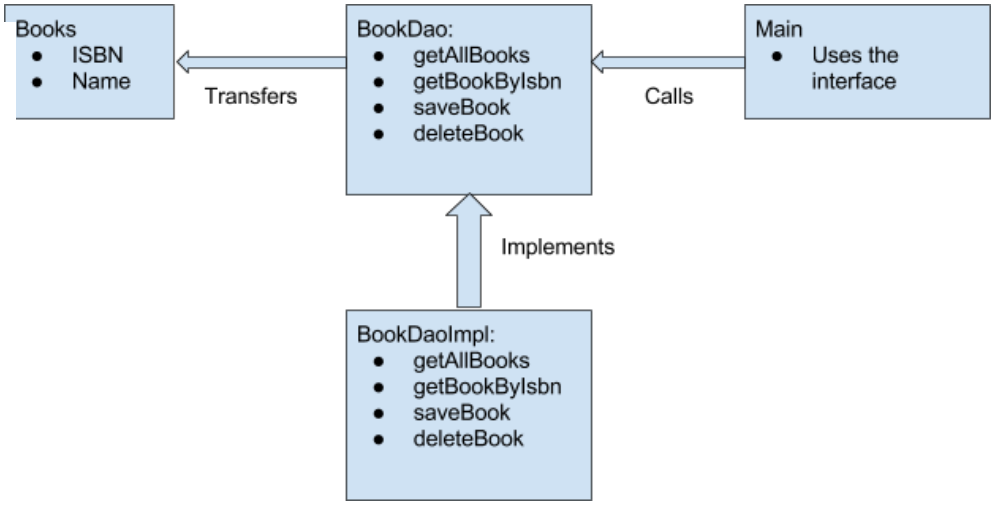
mplementing DAO pattern

With above mentioned components, let's try to implement the DAO pattern. We will use 3 components here:

1. The `Book` model which is transferred from one layer to the other.
2. The `BookDao` interface that provides a flexible design and API to implement.
3. `BookDaoImpl` concrete class that is an implementation of the `BookDao` interface.

Let us put this logic into a diagram:

Tutorials
Java Tutorials
Java EE Tutorials



DAO Pattern model Class

Now, let's put up our model object.

```
package com.journaldev.model;

public class Books {
```

```
    private int isbn;
    private String bookName;

    public Books() {
    }

    public Books(int isbn, String bookName) {
        this.isbn = isbn;
        this.bookName = bookName;
    }

    // getter setter methods
}
```

: is a simple object with just 2 properties to keep things simple.

DAO Pattern Interface

.et's define the interface to access the data associated with it at persistence level.

```
package com.journaldev.dao;

import com.journaldev.model.Books;

import java.util.List;

public interface BookDao {

    List<Books> getAllBooks();
```

```
    Books getBookByIsbn(int isbn);
    void saveBook(Books book);
    void deleteBook(Books book);
}
```

DAO Pattern Implementation

Next, we create a concrete class implementing the above interface.

```
package com.journaldev.daoimpl;

import com.journaldev.dao.BookDao;
import com.journaldev.model.Books;

import java.util.ArrayList;
import java.util.List;

public class BookDaoImpl implements BookDao {

    //list is working as a database
    private List<Books> books;

    public BookDaoImpl() {
        books = new ArrayList<>();
        books.add(new Books(1, "Java"));
```

```
        books.add(new Books(2, "Python"));
        books.add(new Books(3, "Android"));
    }

    @Override
```

Jsing DAO Pattern

Finally, we put this implementation to use in our main() method:

```
package com.journaldev;

import com.journaldev.dao.BookDao;
import com.journaldev.daoimpl.BookDaoImpl;
import com.journaldev.model.Books;

public class AccessBook {

    public static void main(String[] args) {

        BookDao bookDao = new BookDaoImpl();

        for (Books book : bookDao.getAllBooks()) {
            System.out.println("Book ISBN : " +
book.getIsbn());
        }

        //update student
        Books book = bookDao.getAllBooks().get(1);
        book.setBookName("Algorithms");
```

```
        bookDao.saveBook(book);
    }
```

Advantages of DAO pattern

There are many advantages for using DAO pattern. Let's state some of them here:

1. While changing a persistence mechanism, service layer doesn't even have to know where the data comes from. For example, if you're thinking of shifting from using MySQL to MongoDB, all changes are needed to be done in the DAO layer only.
2. DAO pattern emphasis on the low coupling between different components of an application. So, the View layer have no dependency on DAO layer and only Service layer depends on it, even that with the interfaces and not from concrete implementation.
3. As the persistence logic is completely separate, it is much easier to write Unit tests for individual components. For example, if you're using JUnit and Mockito for testing frameworks, it will be easy to mock the individual components of your application.
4. As we work with interfaces in DAO pattern, it also emphasizes the style of "work with interfaces instead of implementation" which is an excellent OOPs style of programming.

DAO Pattern Conclusion

In this article, we learned how we can put DAO design pattern to use to emphasize on keeping persistence logic separate and so, our components loosely coupled.

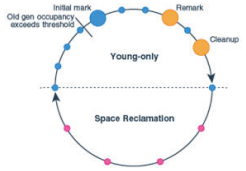
Design patterns are just based on a way of programming and so, is language and framework independent. Feel free to leave your views in comments below. Download the DAO example project from below link.

Download DAO Pattern Example Project

References: Oracle Documentation, Wikipedia.

Java Design Patterns - Example Tutorial

Facade Design Pattern in Java



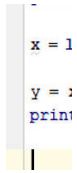
Garbage Collection in Java

Mediat Patterr

Command Design Pattern

Java Dependency Injection - DI Design Pattern Example...

SQL Interview Questions and Answers



Pyth

« PREVIOUS
Factory Design Pattern In Scala

NEXT »
MVC Design Pattern

FILED UNDER: DESIGN PATTERNS

Comments

Rujhan Arora says

May 20, 2019 at 11:19 pm

The class name should be Book, not Books..

Reply

Medy says

April 6, 2019 at 3:25 pm

Thank you for the explanation □

Reply

Sameer says

March 15, 2019 at 12:08 am

Good Explanation on Design patterns and it would be more easy to understand if shown using diagram

Reply

Jane says

March 6, 2019 at 3:16 am

Please correct me if I'm wrong but I'm wondering if there's a typo in "While changing a persistence mechanism, service layer doesn't even have to know where the data comes from. For example, if you're thinking of shifting from using MySQL to MongoDB, all changes are needed to be done in the DAO layer only." I think the changes are needed to be done in the Impl layer instead.

Looking forward to your reply.

Reply

Pankaj says

March 6, 2019 at 8:54 am

DAO interface and its implementation are part of the DAO layer.

Reply

Hemanth Kumar says

May 18, 2018 at 7:35 pm

DAO Design Pattern belongs to which design pattern – Structural or Behavioural DP?

Reply

arun singh says

April 6, 2018 at 8:16 am

thanks

Reply

C S Chakravarthi says

April 4, 2018 at 9:36 pm

Thanks a lot for the explanation

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

POST COMMENT