DAO with Null Object Pattern

Ask Question

After Reading:
Effective Java (See Item 43) - Joshua Bloch
Clean Code (Don't Return Null) - Uncle Bob
Avoiding != null statements
Null Object pattern

I was looking for an answer to the question of what a DAO should return when a search ends up to be for an entity that does not exist for non-collection objects. Collection object is really ok by using empty array or emptyList methods. But with non-collections it might be harder. An alternative solution is to never return null and instead use the Null Object pattern. But I have no idea to integrate with Null Object pattern with DAO and I really excited to see great integration with Null Object pattern and DAO pattern especially for model(dto) object return case.

I would appreciate and welcome any best design pattern, scenario and suggestion.

java nullpointerexception null dao null-object-pattern

edited May 23 '17 at 10:29
Community ♦
1 1

asked Oct 17 '16 at 8:56

Ye Win
1,298 6 16

1 Could returning an `Optional` be helpful? – ChiefTwoPencils Oct 17 '16 at 9:05

You could also throw an exception if you're that set on not using null. – chrylis Oct 17 '16 at 9:06

@ChiefTwoPencils, ya, but i come across from this dzone.com/articles/java-8-optional-avoid-null-and ,as to comments it's not best practice yet and Null Object Pattern is still best idea. – Ye Win Oct 17 '16 at 9:07

@chrylis, I expect for the case where null is a valid response in terms of the contract; and – Ye Win Oct 17 '16 at 9:08

NullObject pattern is returning an object that matches the signature of your populated object (user) with an object that the rest of the code can determine that a null object would have been returned (EmptyUser). The whole idea behind it is that object.getName() will never return null but instead return some entry that you can check ie if(user.getName().equals("e'); drop table Students;") then //we have empty object (or in this case little Bobby drop tables) – Theresa Forster Oct 24 '16 at 12:33

3 Answers

Indeed introducing `null` reference is probably one of the worse mistake in the programming languages' history even its creator **Tony Hoare** calls it his *billion-dollar mistake*.

Here are the best alternatives to `null` according to your `Java` version:

1. `Java 8` and above

Starting from **Java 8** you can use `java.util.Optional` .

Here is an example of how you could use it in your case:

```
public Optional<MyEntity> findMyEntity() {
    MyEntity entity = // some query here
    return Optional.ofNullable(entity);
}
```

2. Prior to `Java 8`

Before **Java 8** you can use `com.google.common.base.Optional` from **Google Guava**.

Here is an example of how you could use it in your case:

```
public Optional<MyEntity> findMyEntity() {
    MyEntity entity = // some query here
    return Optional.fromNullable(entity);
}
```

edited Oct 20 '16 at 19:59

answered Oct 20 '16 at 11:21

Nicolas Filotto

31.3k 8 40 63

dzone.com/articles/java-8-optional-avoid-null-and , I come across this website about Optional but your example is exceptionally good. I would like to know that do you prefer Optional than the Null Object Pattern for null handling case? – Ye Win Oct 21 '16 at 2:18

1 Yes I definitively prefer `Optional` it is much more generic and it has been adopted by Oracle and Google, 2 of the biggest IT companies in the world which give a lot of credits to it. I would even say that `Null Object Pattern` doesn't make any sense anymore in java, it is outdated, `Optional` is the future if you check a little bit what's new in Java 9, you will see that Oracle makes `Optional` go a little bit further, read this – Nicolas Filotto Oct 21 '16 at 9:35

2 The use of Optional, says that the cardinality of the return value is 0 or 1. So it gives a clear description that there could be no value and let the caller decide what to do with it – bobK Oct 25 '16 at 13:29

This is so useful, but i was wondering if we should use Optional in DAO layers too? I mean the return value, should it be Optional<Something>? or should it be in higher layers like services? and by the way how about restful services return value? – Shilan Dec 4 '17 at 13:43

@Shilan my answer is simply why not having it in your DAO interface? – Nicolas Filotto Dec 4 '17 at 14:07

All you need to do is return an empty object - say a customer entry you would have in your DAO something like

if (result == null) { return new EmptyUser(); }

where EmptyUser extends User and returns appropriate entries to getter calls to allow the rest of your code to know it is an empty object (id = -1 etc)

A small example

```
public class User {
    private int id;
    private String name;
    private String gender;
    public String getName() {
       //Code here
    }
```

```
    public void setName() {
       //Code here
    }
}

public class EmptyUser extends User {

    public int getId() {
       return -1;
    }

    public String getName() {
       return String.Empty();
   }
}

public User getEntry() {
   User result = db.query("select from users where id = 1");
   if(result == null) {
       return new EmptyUser();
   }
   else {
       return result;
    }
}
```

edited Oct 20 '16 at 8:50

answered Oct 17 '16 at 8:59

Theresa Forster

1,363 3 13 27

yes, it's null object pattern for this case but could you please provide some pattern diagram and provide full code for EmptyUser(NullObject) class? assume 10 fields had in EmptyUser class. – Ye Win Oct 17 '16 at 9:03

In my experience, in real-life scenarios where a single entity should be returned returning a null is actually either a data inconsistency error or lack of data whasoever. In both cases the *really* good thing to do is to crate and throw your own `DataNotFoudException` . **Fail fast**.

I'm using mybatis as ORM and recently I started writing some theoretically single-result mapper selects as returning lists and validating checking the amount of returned data in dao, and throwing exceptions when the returned amounts do not match dao's method assumptions. It works pretty well.

answered Oct 21 '16 at 7:19

Dariusz
14.8k 5 41 76

Thanks for your answer, but sometimes I expect for the case where null is a valid response, for this case I want to control with custom message and some navigation to users. Especially in combo-box value display and it master setup case. – Ye Win Oct 21 '16 at 7:53

Throwing an exception is still an option. You can catch it anywhere in your stack. But that's for abnormal situations only, so if you really expect your query to return null, then simply go with null or Optional. – Dariusz Oct 21 '16 at 9:51