

## Re: Design Question – formatted this time – sorry

*Source:* <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2004-12/1795.html>

---

*From:* Stephen Riehm (*stephen.riehm\_at\_gmx.net*)

*Date:* 12/15/04

Date: Wed, 15 Dec 2004 22:25:02 +0100

[yuk! Thunderbird didn't word-wrap – sorry]

Hi,

I'm designing a small stand-alone planning tool whose sole purpose is to provide an intuitive way to manipulate data in two tables of a Database (Oracle 9iR2). One table is a list of items to be planned (id, name, duration etc), and the other table contains the planned use of those items. There will never be more than 3 or 4 users and they normally don't work concurrently (they also sit more or less in the same office). Of those users, only 2 of them are allowed to edit the item definitions, but all users need write access to the plan table. Planning is always done on a 24 hour basis, so when one planner starts planing a day, the others should only be allowed to plan other-days.

I was planing on writing this as a client-only program which would use JDBC to connect to the database directly. A check-out table (recording who has been using which data since when) and an oracle sequence is sufficient for simple collision detection. A full-blown client-server architecture for this setup seems like hitting flies with a sledgehammer (since most of the work is done right behind the gui, the complexity and level of distribution. It wont grow more complex either, since the workload is directly related to the number of hours in a day, and nothing else).

The use case would be something like: The user choses a day that they wish to plan, at which point the relevant data for that day would be "checked out" from the database into memory, manipulated off-line and then sent back to the database when the user is finished.

Obviously, a user should not be able to check-out data if the data is already checked out, or if they don't have the priveleges to change that particular set of data.

Which brings my to my question:

are the `java.security.Permissions` and `GuardedObject` suitable for this?

comp.lang.java.programmer: Re: Design Question – formatted this time – sorry

The java.security package seems to be more interested in protection against nasties than simple, friendly access control. Also, GuardedObject only checks the permissions when you try to "run" the object. I want to be able to show the user what they can and can't do BEFORE they try (a bit like greyed out menus etc). I was thinking of getting the user's name from the OS, loading the permissions as a collection of objects from the database and then checking those permissions while putting the GUI together (setting up overview lists, edit buttons etc).

Does anyone have any tips? Am I barking up the wrong tree? Although I'm an experienced programmer, I haven't been writing java for very long and I'm still coming to grips with the API. (It's a "can't see the forest for the trees" problem :-)

Thanks in advance for your guidance!

Steve