

Re: Decouple SQL queries from class in OOP design

Source: <http://coding.derkeiler.com/Archive/General/comp.object/2005-11/msg00177.html>

- *From:* Patrick May <pjm@xxxxxxx>
 - *Date:* 22 Nov 2005 08:42:38 +0000
-

"Mikito Harakiri" <mikharakiri_nospaum@xxxxxxxxxx> writes:
> boolean isValidEmployee(int empNo) {
> //wrap around "select count(*) from emp where emp# = :empNo"
> }
>
> OK, boolean more realistically is a record with picture, first and
> last names fields, but I digress. Tell me a single reason why this
> SQL query shouldn't be embedded in this function.

If the data is owned by the application, embedding SQL might be justified. In most environments of even slightly above trivial levels of complexity, information such as this is maintained in a database that is shared across the enterprise. The reasons why the application should not embed SQL in this case include:

- It couples the application to the database schema. Changes to the schema (due, for example, to requirements of completely different applications or database administrator decisions) could break the application.
- The "validate employee ID" functionality is likely to be used in more than one application. Encapsulating it as a reusable component or service reduces the complexity of each application and provides more consistency across the enterprise.
- Embedded SQL requires more than a simple SQL statement. The application must deal with database connections, transactions, cursors, etc. Encapsulating this overhead and allowing it to be reused across multiple applications reduces the complexity of each application and the opportunity for bugs to be introduced.
- Encapsulating SQL in components or services that expose required functionality instead of the underlying database schema makes impact analysis much more straightforward and reliable. When SQL is embedded in applications across the enterprise, it becomes virtually impossible to determine the effect of any proposed change to the database schema.

Re: Decouple SQL queries from class in OOP design

Note that these issues are independent of the implementation approach. The value of decoupling the application from the database schema exists if your implementation uses procedural, OO, functional, or any other paradigm.

Regards,

Patrick

S P Engineering, Inc. | The experts in large scale distributed OO
| systems design and implementation.
pjm@xxxxxxx | (C++, Java, Common Lisp, Jini, CORBA, UML)

- **Follow-Ups:**

- ◆ **[Re: Decouple SQL queries from class in OOP design](#)**
 - ◇ From: Mikito Harakiri
- ◆ **[Re: Decouple SQL queries from class in OOP design](#)**
 - ◇ From: frebe

- **References:**

- ◆ **[Decouple SQL queries from class in OOP design](#)**
 - ◇ From: Hongyu
 - ◆ **[Re: Decouple SQL queries from class in OOP design](#)**
 - ◇ From: Robert C . Martin
 - ◆ **[Re: Decouple SQL queries from class in OOP design](#)**
 - ◇ From: Mikito Harakiri
 - ◆ **[Re: Decouple SQL queries from class in OOP design](#)**
 - ◇ From: Bruno Desthuilliers
 - ◆ **[Re: Decouple SQL queries from class in OOP design](#)**
 - ◇ From: Mikito Harakiri
- Prev by Date: **[Re: Decouple SQL queries from class in OOP design](#)**
 - Next by Date: **[Looking for a good book on object-oriented GUI programming](#)**
 - Previous by thread: **[Re: Decouple SQL queries from class in OOP design](#)**
 - Next by thread: **[Re: Decouple SQL queries from class in OOP design](#)**
 - Index(es):
 - ◆ **[Date](#)**
 - ◆ **[Thread](#)**