

Re: searching for yoda – a developer's tale

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2005-02/3103.html>

From: none (_at_(none))

Date: 02/24/05

Date: Thu, 24 Feb 2005 17:19:09 -0500

I started like you a bare simple editor, command line tools and should I added hated IDEs at first, reason being I found they are less productive. At least ones I tried couple years ago (VisualAge for Java and JBuilder).

Anyway, at my current job I was exposed to Eclipse and like it very much. A real *IDE* which offer a large number of plugin for almost any thing you want to do, write UML specs, design DB schema, use CVS, want integrated debugging with Servlet/JSP containers to name few.

I also learned SQL first and Java later which resulted in a preference to use simple JDBC. I always felt JDBC allows you more control on actual transactions and does result in some (if done properly) performance boost. However, in a team environment you find some people in team are not familiar or comfortable with SQL hence simple JDBC causes/poses problems. Especially if there are chances of leavening *inconsistent data* especially if object graph is non-trivial with collections. When my manger asked to move to Hibernate I was a little skeptical. But now, I find it almost as flexible and strong as simple JDBC in terms of DB manipulation. That said, there are some points you may have to consider on per project basis, mainly regarding performance.

As for as learning is concerned, I found Eclipse to be intuitive and never read any reference/manual just for the purpose of learning. However, there were times when I need to figure out *how can I do that ...* and for that you can find many on line article. Hibernate is different story, I think "Hibernate in Action" is a good resource.

That said I think there are a number of other interesting technologies out there as well. Eclipse+Hibernate combo is just a small subset for a small problem domain.

Haroon

dexter@uwm.edu wrote:

- > *The mundane problems are with simply writing out the SQL queries to*
- > *include the JOIN clause(s) and not have any typos. Typos show up at*
- > *runtime and the error reported only gives you the line of the JDBC*
- > *call, not the particular SQL bit in question. This is tedious and*

- > *annoying, but by itself probably wouldn't force me to seek learning. A*
- > *similar problem is that with "standard" SQL there isn't an*
- > *INSERT_OR_UPDATE, and the format for said commands differs so one must*
- > *do a SELECT followed by an if (exists) UPDATE else INSERT. Effectively*
- > *writing three statements to perform one task. If there are many items*
- > *in your object graph that could be inserted/updated, the tedium (and*
- > *typos) increases.*
- >
- > *The real problem though is worrying about persistence of object graphs.*
- > *If I have a persistent object that has a collection of objects, then*
- > *the UPDATE statement (or really, the INSERT_OR_UPDATE bit) for the main*
- > *object needs to check each of the items in the collection to see if*
- > *they've changed (do an UPDATE), been deleted (do a DELETE), or been*
- > *added (do an INSERT). Suppose the relationship is bidirectional.*
- > *Suppose the collection is ordered (a List). One needs to pay close*
- > *attention to what the book "Hibernate In Action" terms "the scope of*
- > *object identity" -- what does it mean to say "this object has already*
- > *been saved, but has changed"? It implies you need your code to*
- > *maintain database keys which it shouldn't have to do.*
- >
- > *All of these things end up making me have to think too hard about stuff*
- > *that the folks in relational db land have understood for decades. I*
- > **believe* that Hibernate in particular has been able to shield the*
- > *developer from much of this. We work with objects. The*
- > *object/relational mapping problem is non-trivial and has been thought*
- > *about and implemented by folk with more time (and likely bigger brains)*
- > *than me -- and they've put their stuff out there for any to use....just*
- > *wish I knew how to take full advantage of it.*
- >
- > *-don.*
- >