Research Centers
Core Java
Enterprise Java
Mobile Java
Tools & Methods
JavaWorld Archives

Site Resources
Featured Articles
News & Views
Community
Java Q&A
JW Blogs
Podcasts
Site Map
Careers
Newsletters
Whitepapers
RSS Feeds

About JavaWorld
Advertise
Write for JW

iBATIS, Hibernate, and JPA: Which is right for you?

Object-relational mapping solutions compared

By K. L. Nitin, Ananya S., Mahalakshmi K., and S. Sangeetha, JavaWorld.com, 07/15/08

Object-relational mapping

Object-relational mapping (ORM) has emerged as a solution to what is sometimes called the object-relational impedance mismatch. ORM is a technique that transparently persists application objects to the tables in a relational database. ORM behaves like a virtual database, hiding the underlying database architecture from the user. ORM provides functionality to perform complete CRUD operations and encourages object-oriented querying. ORM also supports metadata mapping and helps in the transaction management of the application.

An example will help illustrate how ORM works. Consider a simple `Car` object that needs to be persisted in the database. The `Car` object in the domain model is the representation of the CAR table in the data model. The attributes of the `Car` object are derived from the columns of the CAR table. There is a direct mapping between the `Car` class and the CAR table, as illustrated in Figure 1.

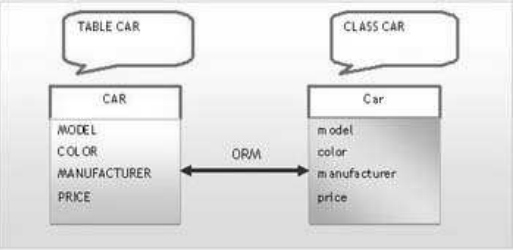

Figure 1. Mapping an object to a table

There are many open source ORM tools, including Hibernate, iBATIS SQL Maps, and Java Ultra-Lite Persistence. Most of these tools are *persistence frameworks* that provide a layer of abstraction between the Java application and the database. A persistence framework maps the objects in the application domain to data that needs to be persisted in a database. The mappings can be defined using either XML files or metadata annotations (the latter introduced to the language as part of Java 1.5). The persistence framework aims to separate the database-related code and the application code (that is, the business logic), thereby increasing application flexibility. A persistence framework simplifies the development process by providing a wrapper around the persistence logic.

With this basic introduction to persistence out of the way, we're ready to move on to discussing two of the most popular open source persistence frameworks, iBATIS and Hibernate. We'll also introduce the Java Persistence API and discuss the strengths and weaknesses of all three solutions in various application scenarios

JW's Most Read
Scala on their minds -- bloggers debate Scala's complexity
Simplify the data access layer with Spring 3.1 and Java generics
Bitcoin for beginners, Part 2: Bitcoin as a technology and network
Why Yammer dropped Scala for Java
Engine Yard vs Heroku: Ruby clouds (for Java too)

Featured White Papers
Observation Driven Testing: What else is your code doing?
Software Agitation: Your Own Personal Code Reviewer
Early & Often: Avoiding Security Flaws with CI, High Coverage

scenarios, however, you will need more direct control over SQL queries. When writing an application that involves a series of update queries, it's more effective to write your own SQL queries than to rely on ORM-generated SQL. Also, ORM cannot be used when there is a mismatch between the object model and the data model. As we've mentioned, JDBC code was once the common solution to such problems, but it introduced a lot of database code within application code, making applications harder to maintain. A persistence layer is needed to decouple the application and the database.

The iBATIS Data Mapper framework helps solve these problems. iBATIS is a persistence framework that provides the benefits of SQL but avoids the complexity of JDBC. Unlike most other persistence frameworks, iBATIS encourages the direct use of SQL and ensures that all the benefits of SQL are not overridden by the framework itself.

iBATIS in brief
The iBATIS project was initiated by Clinton Begin and released in 2001. This persistence framework was initially designed for Java, though it has since been extended to support other platforms, including .Net and Ruby.

Simplicity is iBATIS's greatest advantage, as it provides a simple mapping and API layer that can be used to build data-access code. In this framework the data model and the object model need not map to one another precisely. This is because iBATIS uses a *data mapper*, which maps objects to stored procedures, SQL statements, or `ResultSet`s via an XML descriptor, rather than a *metadata mapper*, which maps objects in the domain to tables in the database. Thus, iBATIS enables the data model and the object model to be independent of each other.

How iBATIS works

iBATIS allows loose coupling of the database and application by mapping the input to and output from the database to the domain objects, thus introducing an abstraction layer. The mapping is done using XML files that contain SQL queries. This loose coupling allows the mapping to work for systems where the application and the database design are mismatched. It also helps in dealing with legacy databases and with databases that change over time.

The iBATIS framework mainly uses the following two XML files as descriptors:

- SQLMapConfig.xml
- SQLMap.xml

We'll look at each file in detail.

SQLMapConfig.xml

SQLMapConfig.xml is a central XML file that contains all the configuration details, like the details for the data sources; it also optionally includes information about transaction management. This file identifies all the SQLMap.xml files -- of which there may be more than one -- and loads them.

Consider an `Employee` class that maps to an EMPLOYEE table in the database. The properties of the class -- `emp_id`, `emp_firstname`, and `emp_lastname` -- correspond to similarly named columns in the table. The class diagram for the `Employee` class is shown in Figure 2. (This class will be used to demonstrate the different persistence techniques that are discussed in this article.)

Figure 2. Class diagram for the Employee class

The SQLMapConfig.xml file for the `Employee` class can be written as shown in Listing 1.

Listing 1. SQLMapConfig.xml file for Employee

```
<sqlMapConfig>
  <transactionManager type="JDBC" commitRequired="false">
    <dataSource type="EMPLOYEE">
     <property name="JDBC.Driver"           value="com.mysql.jdbc.Driver"/>
      <property name="JDBC.ConnectionURL" value="jdbc:mysql://localhost:3306/ibatis"/>
      <property name="JDBC.Username" value="root"/>
      <property name="JDBC.Password" value=""/>
    </dataSource>
  </transactionManager>
 <!-- List the SQL Map XML files. They can be loaded from the classpath, as they are here (com.mydomain.data...
  <sqlMap resource="com/mydomain/data/Employee.xml"/>
 </sqlMapConfig>
```

SQLMapConfig.xml uses a `transactionManager` tag to configure a data source to use with this particular SQL map. It specifies the type of the data source, along with some details, including information about the driver, the database URL, and the username and password. The `sqlMap` tag specifies the location of the SQLMap.xml file so as to load it.

Add New Comment

Showing 20 of 24 comments

Sort by newest first

Ahamed Firzadh

Hats off gr8 article.... Thanks for the detailed explanation which is also easy on a newbie :)

11/21/2011 06:14 AM Like Reply

Anonymous

IBatis, Hibernate, JPA
Thank you for putting this together. It provides a very good introduction for me into ORM and persistence. Much appreciated.

06/26/2010 04:39 PM Like Reply

Anonymous

Thanks!!!
Thanks a lot. Very useful article.

05/30/2010 05:34 AM Like Reply

Anonymous

Buen inicio
Muchas gracias por el artículo, me dió lo necesario para empezar :)

05/22/2010 11:06 PM Like Reply

Anonymous

good article.
Thanks

05/19/2010 02:14 AM Like Reply

Anonymous

Thanks!
Great article!

04/20/2010 04:10 AM Like Reply

Anonymous

Nice comparison
I was searching for a comparison between Hibernate and JPA. Thanks for the effort.

-Sunil.
http://techmindviews.blogspot....

03/26/2010 04:58 AM Like Reply

Anon mo s

the Java World.

03/24/2010 02:21 AM Like Reply

Anonymous

great article and excellent explanation.

03/13/2010 09:21 AM Like Reply

Anonymous

Question
The article mentions that JPA is non-portable. But woudn't it be possible to use JPA in Rails using JRuby?

01/30/2010 08:51 AM Like Reply

Anonymous

Yes,It'a grate
Can i say that,If we want to separated Business layer and database layer that time,we can use this method....

01/22/2010 02:03 AM Like Reply

Anonymous

Hibernate Vs IBatis
IBatis is considered to be the way if you want to have full control over your queries + dont want to clutter your code with sqls.

I would contest that by saying hibernate provides the exact same feature with the concept of named queries if you ever needed that feature.

The only reason i think people tend to tilt towards IBatis is that it does not overwhelm you with tons of features which first time users find a bit daunting and get scared away.

01/21/2010 01:04 PM Like Reply

Anonymous

This is it.
I was searching for database layer solutions. This article helped me a lot. I am enlightened. Thank you so much.

12/06/2009 09:45 PM Like Reply

Anonymous

Control over queries...
"Hibernate provides a complete ORM solution, but offers you no control over the queries"

Dude, Session.createSQLQuery lets you use native sql, if that is not "full control over queries" then iBATIS does not have it either! PD: RTFM

10/08/2009 02:36 PM Like Reply

Anonymous

Hibernate also lets you custom code sql mappings for its ORM use just like ibatis does too. so if you want/need to code your own sql you can!
you can also configure it to use stored procs too.
ref http://docs.jboss.org/hibernat...

03/24/2010 06:13 PM in reply to Anonymous Like Reply

Anonymous

Question for the authors
Excellent article, I can see a great research work behind.

I'm just nearly convinced to use iBATIS, but I'd like to double check with you my scenario. I'm using an existing database with loads of functions and stored procedures that I'll have to use in my application, that's why I choose iBATIS. On the other hand, I'll have control over both the Java and

provided by Hibernate may be a better option?

Thanks a lot!

10/06/2009 08:21 AM Like Reply

Anonymous

iBatis is my choice - from experience...
There's no silver bullet, so use the technology you are most familiar with or standard in the environment. BUT, having used iBatis and hibernate extensively, I always default to iBatis. I have spent too many hours trying to figure out what hibernate is doing - as performance can be an issue. I rarely work on a project that needs to access more than one database type (mysql, oracle, sybase, ms sql server, postgres etc).

I have just finished a .Net desktop project with iBatis as the db access layer. It works a treat!

A large financial organisation performed extensive research on java tools for their global architecture team ... and iBatis came out on top for the db access layer.

So if its a new project, give iBatis a try.

09/02/2009 12:26 AM Like Reply

Anonymous

Hibernate's Simplicity is "Good"? Bah...
How on Earth did you conclude that Hibernate's simplicity is 'good'? Let me tell you why it should be downgraded to 'average' if not 'poor'. The official Hibernate book is huge, and the number of times we've had to research how to solve a problem on our project is ridiculous. Ramp up time is long for new teammates. The reason why it seems community support is so good for Hibernate is because we see so many posts of people have problems with the technology.

08/03/2009 11:34 PM Like Reply

Anonymous

it is a very helpful article and also has a very simple manner of telling. I was not bored while reading.
Thank you so much..

07/17/2009 12:39 AM Like Reply

Anonymous

Hibarnate seems to be the best
Hibernate seems to be winner

07/09/2009 05:28 AM 1 Like Like Reply

Subscribe by email RSS

Load more comments

Archived Discussions (Read only)

Forum migration complete *By Athen*

Forum migration update *By Athen*

Resources

Learn more about the three technologies discussed in this article from the project homepages:

- Hibernate
- iBATIS
- The Java Persistence API

"Get started with Hibernate" (Christian Bauer and Gavin King, JavaWorld, October 2004) is a short introduction to Hibernate written by its creator, Gavin King. *Excerpted from Hibernate in Action; Manning 2004.*)

(Roland Barcia, Geoffrey Hambrick, Kyle Brown, Robert Peterson, Kulvir Singh Bhogal; IBM Press, May 2008).

Ted Neward introduces the so-called object-relational impedance mismatch in his blog post "The Vietnam of computer science."

"Flexible reporting with JasperReports and iBatis" (Scott Monahan, JavaWorld, December 2007) is a hands-on introduction to the iBatis Data Mapper framework.

"Understanding the Java Persistence API" (Aditi Das, JavaWorld, January 2008) is a two-part introduction to Java-platform persistence with OpenJPA.

Java-source.net lists a roundup of open source persistence frameworks for Java.

Visit the JavaWorld Java Enterprise Edition research center for more articles about enterprise data management and Java persistence solutions.

Also see Network World's IT Buyer's Guides: Side-by-side comparison of hundreds of products in over 70 categories.