



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
New sletters
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

Print Feedback 24 Comments Like 37

Object-relational mapping in Java is a tricky business, and solutions like JDBC and entity beans have met with less than overwhelming enthusiasm. But a new generation of ORM solutions has since emerged. These tools allow for easier programming and a closer adherence to the ideals of object-oriented programming and multi-tiered architectural development. Learn how Hibernate, iBATIS, and the Java Persistence API compare based on factors such as query-language support, performance, and portability across different relational databases.

In this article we introduce and compare two of the most popular open source persistence frameworks, iBATIS and Hibernate. We also discuss the Java Persistence API (JPA). We introduce each solution and discuss its defining qualities, as well as its individual strengths and weaknesses in broad application scenarios. We then compare iBATIS, Hibernate, and JPA based on factors such as performance, portability, complexity, and adaptability to data model changes.

If you are a beginning Java programmer new to persistence concepts, reading this article will serve as a primer to the topic and to the most popular open source persistence solutions. If you are familiar with all three solutions and simply want a straightforward comparison, you will find it in the section "Comparing persistence technologies."

Understanding persistence

Persistence is an attribute of data that ensures that it is available even beyond the lifetime of an application. For an object-oriented language like Java, persistence ensures that the state of an object is accessible even after the application that created it has stopped executing.

There are different ways to achieve persistence. The traditional approach to the problem is to use file systems that store the necessary information in flat files. It is difficult to manage large amounts of data in this way because the data is spread across different files. Maintaining data consistency is also an issue with flat-file systems, because the same information may be replicated in various files. Searching for data in flat files is time-consuming, especially if those files are unsorted. Also, file systems provide limited support for concurrent access, as they do not ensure data integrity. For all these reasons, file systems are not considered a good data-storage solution when persistence is desired.

The most common approach today is to use databases that serve as repositories for large amounts of data. There are many types of databases: relational, hierarchical, network, object-oriented, and so on. These databases, along with their database management systems (DBMSs), not only provide a persistence facility, but also manage the information that is persisted.

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

Newsletter sign-up [View all new sletters](#)

Enterprise Java Newsletter

Stay up to date on the latest tutorials and Java community news posted on JavaWorld

Sponsored Links

Slow Apps? Overloaded DBMS? - FREE Download
FREE Download Feb 14 - March 7th Eliminate bottlenecks & maximize performance ScaleOut StateServer®

Sponsored Links

Optimize with a SATA RAID Storage Solution
Range of capacities as low as \$1250 per TB.
Ideal if you currently rely on servers/disks/JBODs

presentation, business, and database-related code into different tiers (or *layers*) of the application. The layer that separates the business logic and the database code is the *persistence layer*, which keeps the application independent of the underlying database technology. With this robust layer in place, the developer no longer needs to take care of data persistence. The persistence layer encapsulates the way in which the data is stored and retrieved from a relational database.

Java applications traditionally used the JDBC (Java Database Connectivity) API to persist data into relational databases. The JDBC API uses SQL statements to perform create, read, update, and delete (CRUD) operations. JDBC code is embedded in Java classes -- in other words, it's tightly coupled to the business logic. This code also relies heavily on SQL, which is not standardized across databases; that makes migrating from one database to another difficult.

Relational database technology emphasizes data and its relationships, whereas the object-oriented paradigm used in Java concentrates not on the data itself, but on the operations performed on that data. Hence, when these two technologies are required to work together, there is a conflict of interests. Also, the object-oriented programming concepts of inheritance, polymorphism, and association are not addressed by relational databases. Another problem resulting from this mismatch arises when user-defined data types defined in a Java application are mapped to relational databases, as the latter do not provide the required type support.

Print

Feedback

Like

37

Like

Comments powered by

Our Commenting Network | Policies | Privacy


Add New Comment

Login



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



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



Anonymous

Hibernate Vs IBatis
The article is awesome for the beginners. Thanks for the valuable information that you have put on 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 wouldn't it be possible to use JPA in Rails using JRuby?

01/30/2010 08:51 AM

Like Reply



Anonymous

Yes, it's a grate
Can i say that, If we want to separate 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 + don't 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



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 the DB; but no time to migrate the Stored Procedures to Java.

Performance will be a very important issue, as it's an application which will manage loads of users and heavy binary and xml stored data. I can tune the queries on iBATIS, but the cache mechanism 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

Hibernate seems to be the best
Hibernate seems to be winner

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

✉ Subscribe by email RSS

Load more comments

...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.*

Java Persistence with Hibernate (Christian Bauer and Gavin King; Manning, November 2006) is the updated second edition of *Hibernate in Action*. Also see iBATIS in Action (Clinton Begin, Brandon Goodin, and Larry Meadors, 2007).

For broader coverage of Java persistence solutions including JDBC, OpenJPA, and pureQuery, see Persistence in the Enterprise: A Guide to Persistence Technologies (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.



Sponsored Links

Web hosting reviews for you to choose a better host
ManageEngine: End-to-End Java Performance Management. Download Product Now!

RESEARCH CENTERS		IDG Network		
Core Java		CFOworld	GamePro	IDG Ventures
Enterprise Java		CIO	Games.net	InfoWorld
Mobile Java		Computerworld	IDG Connect	ITworld
Tools & Methods		CSO	IDG Knowledge Hub	JavaWorld
JavaWorld Archives		DEMO	IDG TechNetwork	LinuxWorld
				Macworld
				Network World
				PC World