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

Working with Hibernate

When a `SessionFactory` instance is created in the application, Hibernate reads the configuration file and identifies the respective mapping file. The session object that is created from the `SessionFactory` gets a particular connection to the database, and this session object is the persistence context for the instance of a persistence class. The instance can be in one of the three states: *transient*, *persistent*, or *detached*. In the transient state, the object is yet to be associated with a table; in the persistent state, the object is associated with the table; and in the detached state, there is no guarantee that the object is in sync with the table. The Hibernate code that is used to persist an `Employee` object is shown in Listing 7.

Listing 7. Persisting an object with Hibernate

```
Session session = null;
Transaction tx = null;

// At this point the Configuration file is read
SessionFactory sessionFactory = new Configuration().configur

// A specific session object is obtained
session = sessionFactory.openSession();

// A new database transaction is started
tx = session.beginTransaction();

// Employee Object is created & populated
Employee emp = new Employee();
emp.setId(1);
emp.setEmpFirstname("K L");
emp.setEmpLastname("Nitin");

// Using the session, emp object is persisted in the databas
session.save(emp);
```

The mapping file that the configuration file identifies maps a particular persistent class to the database table. It maps specific columns to specific fields, and has associations, collections, primary key mapping, and ID key generation mechanisms. The mapping files are generally given names based on the tables to which they map; in the example application, you'd use Employee.hbm.xml for the file that corresponds to the EMPLOYEE table. As you can see in Listing 8, the mapping file specifies that the `Employee` class has to be mapped to the EMPLOYEE table in the database, which has columns named `id`, `emp firstname`, and

```
<hibernate-mapping package="demo">
  <class name="Employee" table="employee" >
    <meta attribute="sync-DAO">false</meta>
      <id name="Id" type="integer" column="emp_id">
        <generator class="assigned"/>
      </id>
       <property name="EmpFirstname" column="emp_firstname"
              type="string" not-null="true" length="30" />
      <property name="EmpLastname" column="emp_lastname"                      type="string" not-null="true" len
    </class>
</hibernate-mapping>
```

When to use Hibernate

Hibernate is best used to leverage end-to-end OR mapping. It provides a complete ORM solution, but leaves you control over queries. Hibernate is an ideal solution for situations where you have complete control over both the application and the database design. In such cases you may modify the application to suit the database, or vice versa. In these cases you could use Hibernate to build a fully object-relational application. Hibernate is the best option for object-oriented programmers who are less familiar with SQL.

The Java Persistence API

The Java Persistence API is the standard object-relational mapping and persistence management interface for the Java EE 5 platform. As part of the EJB 3 specification effort, it is supported by all major Java vendors. The Java Persistence API draws on ideas from leading persistence frameworks and APIs, such as Hibernate, Oracle TopLink, Java Data Objects (JDO), and EJB container-managed persistence. JPA provides a platform on which specific implementations of persistence providers can be used. One of the main features of the Java Persistence API is that any persistence provider can be plugged in to it.

JPA is a POJO-based standard persistence model for ORM. It is part of the EJB 3 specification and replaces entity beans. The entity beans defined as part of the EJB 2.1 specification had failed to impress the industry as a complete persistence solution for several reasons:

- Entity beans are heavyweight components and are tightly coupled to a Java EE server. This makes them less suitable than lightweight POJOs, which are more desirable for their reusability.
- Entity beans are difficult to develop and deploy.
- BMP entity beans force you to use JDBC, while CMP entity beans are highly dependent on the Java EE server for their configuration and ORM declaration. These restrictions will affect the performance of the application.

To address these issues, the EJB 3 software expert group developed JPA as part of JSR 220. JPA borrows the best ideas from other persistence technologies. It defines a standard persistence model for all Java applications. JPA can be used as the persistence solution for both Java SE and Java EE applications.

JPA uses metadata annotations and/or XML descriptor files to configure the mapping between Java objects in the application domain and tables in the relational database. JPA is a complete ORM solution and supports inheritance and polymorphism. It also defines an SQL-like query language, JPQL (Java Persistence Query Language), which is different from EJB-QL (EJB Query Language), the language used by entity beans.

Hibernate and JPA

Having just finished learning about how Hibernate can serve as a standalone persistence solution, you may be surprised to discover that it can also work with JPA. Strictly speaking, if you're going to use Hibernate by itself, you'll be using the *Hibernate Core* module, which generates SQL using HQL without the need for handling JDBC objects; the application is still independent of databases. Hibernate Core can be used with any application server, and for any generic Java application that needs to perform object-relational mapping. This mapping will is achieved by using native Hibernate APIs, the Hibernate Query Language, and XML mapping.

The Hibernate team was deeply involved in the development of the EJB 3 specification. After the introduction of EJB 3, a standalone implementation of EJB 3 persistence was made available as part of Hibernate -- Hibernate Annotations and Hibernate EntityManager. These two are built on top of Hibernate Core. For applications developed using Java EE 5 in which there is a need to use EJB 3, Hibernate EntityManager can be considered as an option for the persistence provider. Applications developed using Java EE 5 will utilize Hibernate and JPA working together.

With JPA, you can plug in any persistence provider that implements the JPA specification instead of using whatever default persistence provider comes with your Java EE container. For example, the GlassFish server uses TopLink Essentials, provided by Oracle, as its default persistence provider. But you could choose to use Hibernate as the persistence provider instead by including all the necessary JAR files in your application.

< Prev 1 2 3 4 5 6 7 Next >

Print Feedback Like

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

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

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

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 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

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.*)

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).

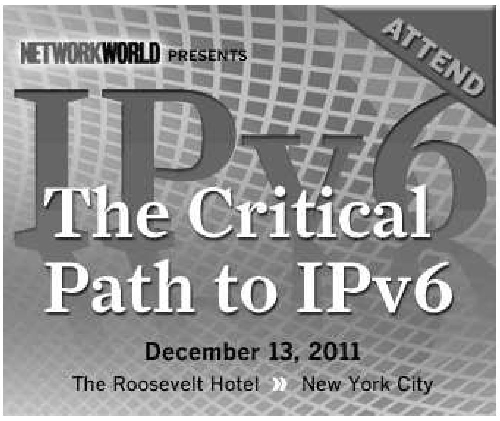
For broader coverage of Java persistence solutions including JDBC, OpenJPA, and pureQuery, see Persistence in the Enterprise: A Guide to Persistence Technologies (Roland Barcia, Geoffrey Hambrick, Kyle Brown, Robert Peterson, Kulvir Singh Bhogal; IBM Press, May 2008).

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

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


NETWORKWORLD PRESENTS
ATTEND
IPv6
The Critical Path to IPv6
December 13, 2011
The Roosevelt Hotel » New York City

Sponsored Links

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