Hibernate ORM

Getting started

Your relational data. Objectively.

Latest stable (7.0)

Idiomatic persistence for Java and relational databases.

Object/Relational Mapping

Hibernate makes relational data visible to a program written in Java, in a natural and type-safe form,

- making it easy to write **complex queries** and work with their results,
- letting the program easily **synchronize changes** made in memory with the database,
- respecting the ACID properties of **transactions**,
- and allowing **performance optimizations** to be made after the basic persistence logic has already been written.

Hibernate is the most successful ORM solution ever, in any programming language.

Unfamiliar with the notion of object/relational mapping? Read more here.

Jakarta Persistence (JPA) compatibility

Right alongside its own "native" API, Hibernate provides an implementation of the Jakarta Persistence specification. In fact, the first version of JPA was inspired by the success of Hibernate 2, and Hibernate was the major influence on the specification and API.

Thus, Hibernate can be easily used in any environment supporting JPA: Java SE standalone applications, EE servers like WildFly, and other containers including Quarkus.

Idiomatic persistence

Latest news

Hibernate 7 (and Hibernate Validator 9)

2025-05-20

Today—or rather, late last night—we released Hibernate ORM 7, which includes the latest version of Hibernate Data Repositories. We've also released Hibernate Validator 9. Hibernate ORM 7...

Hibernate 7.0.0.Final

2025-05-19

Hibernate ORM 7.0 Final has just been released. Details about the release, as well as links to important resources, can be found at https://github.com/hibernate/hib orm/releases/tag/7.0.0 ...

Hibernate 7.0.0.CR2

2025-05-15

A second CR for Hibernate ORM 7.0 has just been released. We cleaned up a few APIs and made some improvements to incubating contracts. Details...

Hibernate 7.0.0.CR1

The project motto is: **relational persistence for idiomatic Java**.

Hibernate lets you write persistent classes following natural object-oriented idioms including inheritance and polymorphism, representing associations using standard Java collections. Hibernate does not require a persistent class to implement an interface, extend a base class, or depend on any other intrusive framework-specific API.

Information about object/relational mappings may be provided by annotating the persistent class with annotations defined by Jakarta Persistence, or by supplying mappings in XML.

Tunable performance

Hibernate provides flexible data fetching strategies, and an extremely-tunable two-level caching architecture. For managing concurrent data access, Hibernate defaults to optimistic locking with automatic versioning or time-stamping, but pessimistic locks may be explicitly requested. Most SQL is generated at system initialization time instead of at runtime.

Hibernate consistently achieves superior performance compared to hand-written JDBC code, in both runtime efficiency and developer productivity. Importantly, Hibernate makes it much easier to tune the performance of code *after* it's written.

Extreme scalability and robustness

From its very birth more than two decades ago, Hibernate was designed specifically for use in highly concurrent environments and in server clusters. Since then, its architecture has been proven via heavy production usage in innumerable enterprise systems covering every major industry, and by its acceptance among hundreds of thousands of Java developers across more than two decades.

Powerful query language

Hibernate Query Language (HQL) is an incredibly powerful object-oriented dialect of SQL, with almost every feature of ANSI SQL `SELECT` statements you've ever heard of. Arguably, it supports more of ANSI SQL than most relational databases.

Since Hibernate Processor has access to your entities and their mapping annotations at compilation time, it's able to completely validate the syntax and typing of a HQL query when you compile your Java code. Hibernate Tools and IntelliJ IDEA even provide automatic code completion and validation for HQL queries as you type.

If all else fails, Hibernate lets you write your own SQL queries in the native dialect of your database.

2025-04-24

A first (and hopefully last, obviously) CR for Hibernate ORM 7.0 has just been released. Details about the release, as well as links to important...

Doing away with SELECT NEW

2025-03-30

The JPQL select new syntax originated in the earliest days of Hibernate, providing a way to package a query projection list into a Java object. This...

Hibernate ORM Moves to Apache License

2025-03-14

The Hibernate team is ecstatic to report that we have successfully relicensed Hibernate ORM under the Apache 2.0 license! As discussed previously, this decision was solely...

Other news →

Compatibility with a wide range of databases

Hibernate is tested every day on PostgreSQL, MySQL, Db2, Oracle, SQL Server, Sybase ASE, EDB, TiDB, MariaDB, HANA, CockroachDB, H2, and HSQLDB. There's even built-in support for Spanner.

The Community Dialects module defines compatibility with a range of older databases, including Informix, Ingres, and Teradata, along with incubating support for SQLite.

Developer joy

Adoption of Hibernate was originally driven by grassroots Java developers looking for a way out of the quagmire of handwritten object/relational mapping code, in-house ORM frameworks, and failed industry standards like entity beans. Later, Hibernate was the first major Java development tool to move from XML to annotations—at a time when other frameworks were still promoting the heavy use of XML for just about any task—and this was a critical step in improving the productivity of Java enterprise development as a whole.

The most recent major versions of Hibernate have placed a central focus on improved error reporting and compile-time type safety, leading to a great improvement in developer experience. This work has driven changes in the latest generation of Jakarta specifications, and in particular lead to the new programming model in Jakarta Data.