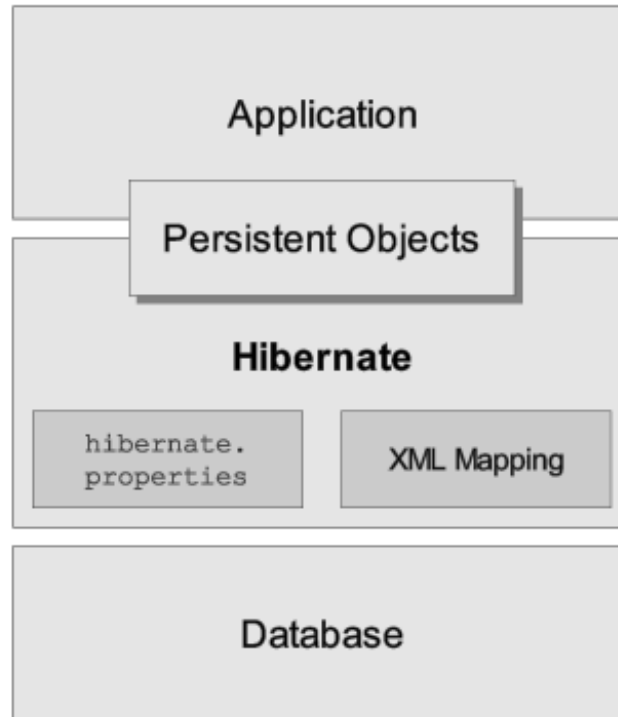


Hibernate Architecture Overview

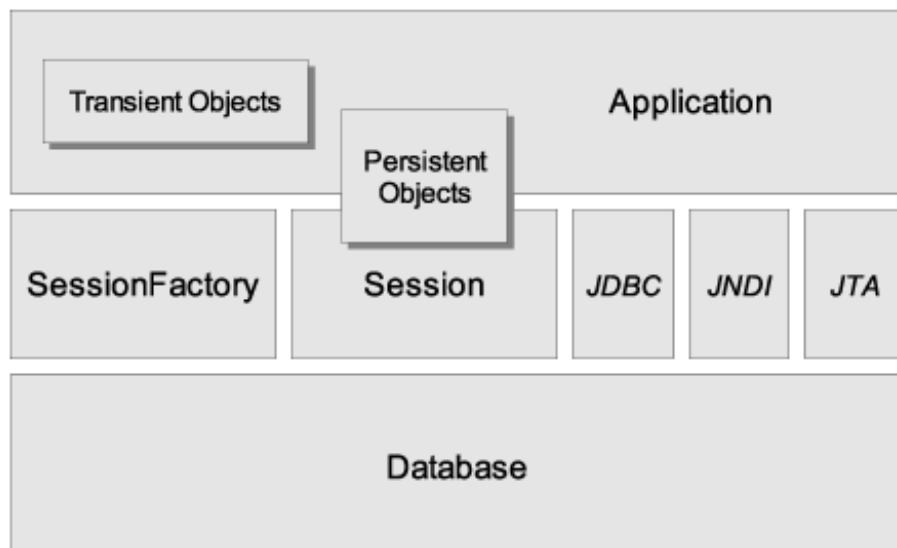
 javacodegeeks.com/2017/06/hibernate-architecture-overview.html

The diagram below provides a high-level view of the Hibernate architecture:

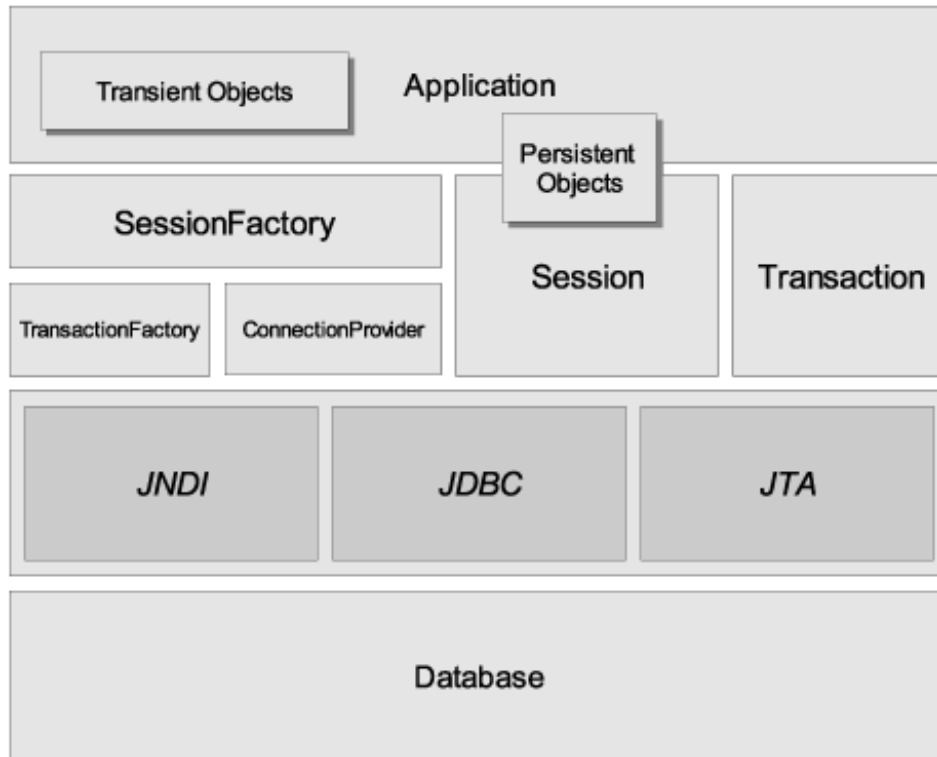


Minimal architecture

The “minimal” architecture has the application manage its own JDBC connections and provide those connections to Hibernate; additionally the application manages transactions for itself. This approach uses a minimal subset of Hibernate APIs.



Comprehensive architecture



Hibernate Basic APIs

Here are quick discussions about some of the API objects depicted in the preceding diagrams.

SessionFactory (`org.hibernate.SessionFactory`)

A thread-safe, immutable cache of compiled mappings for a single database. A factory for `org.hibernate.Session` instances. A client of `org.hibernate.connection.ConnectionProvider`. Optionally maintains a second level cache of data that is reusable between transactions at a process or cluster level.

Session (`org.hibernate.Session`)

A single-threaded, short-lived object representing a conversation between the application and the persistent store. Wraps a JDBC `java.sql.Connection`. Factory for `org.hibernate.Transaction`. Maintains a first level cache of persistent the application's persistent objects and collections; this cache is used when navigating the object graph or looking up objects by identifier.

Persistent objects and collections

Short-lived, single threaded objects containing persistent state and business function. These can be ordinary JavaBeans/POJOs. They are associated with exactly one `org.hibernate.Session`. Once the `org.hibernate.Session` is closed, they will be detached and free to use in any application layer (for example, directly as data transfer objects to and from presentation).

Transient and detached objects and collections

Instances of persistent classes that are not currently associated with a `org.hibernate.Session`. They may have been instantiated by the application and not yet persisted, or they may have been instantiated by a closed `org.hibernate.Session`.

Transaction (`org.hibernate.Transaction`)

(Optional) A single-threaded, short-lived object used by the application to specify atomic units of work. It abstracts the application from the underlying JDBC, JTA or CORBA transaction. A `org.hibernate.Session` might span several `org.hibernate.Transactions` in some cases. However, transaction demarcation, either using the underlying API or `org.hibernate.Transaction`, is never optional.

ConnectionProvider (`org.hibernate.connection.ConnectionProvider`)

(Optional) A factory for, and pool of, JDBC connections. It abstracts the application from underlying `javax.sql.DataSource` or `java.sql.DriverManager`. It is not exposed to application, but it can be extended and/or implemented by the developer.

TransactionFactory (`org.hibernate.TransactionFactory`)

(Optional) A factory for `org.hibernate.Transaction` instances. It is not exposed to the application, but it can be extended and/or implemented by the developer.

Extension Interfaces

Hibernate offers a range of optional extension interfaces you can implement to customize the behavior of your persistence layer. See the API documentation for details.

Reference: [Hibernate Architecture Overview](#) from our [JCG partner](#) Annamalai Thangaraj at the [Java Tutorials Corner](#) blog.
