**Optional configuration properties**

There are a number of other properties that control the behavior of Hibernate at runtime. All are optional and have reasonable default values.

**Warning**

*Some of these properties are "system-level" only.* System-level properties can be set only via java -Dproperty=value or hibernate.properties. They *cannot* be set by the other techniques described above.

**Hibernate Configuration Properties**

| **Property name** | **Purpose** |
| --- | --- |
| hibernate.dialect | The classname of a Hibernate org.hibernate.dialect.Dialect which allows Hibernate to generate SQL optimized for a particular relational database.  **e.g.** full.classname.of.Dialect  In most cases Hibernate will actually be able to choose the correct org.hibernate.dialect.Dialect implementation based on the JDBC metadata returned by the JDBC driver. |
| hibernate.show\_sql | Write all SQL statements to console. This is an alternative to setting the log category org.hibernate.SQL to debug.  **e.g.** true | false |
| hibernate.format\_sql | Pretty print the SQL in the log and console.  **e.g.** true | false |
| hibernate.default\_schema | Qualify unqualified table names with the given schema/tablespace in generated SQL.  **e.g.** SCHEMA\_NAME |
| hibernate.default\_catalog | Qualifies unqualified table names with the given catalog in generated SQL.  **e.g.** CATALOG\_NAME |
| hibernate.session\_factory\_name | The org.hibernate.SessionFactory will be automatically bound to this name in JNDI after it has been created.  **e.g.** jndi/composite/name |
| hibernate.max\_fetch\_depth | Sets a maximum "depth" for the outer join fetch tree for single-ended associations (one-to-one, many-to-one). A 0 disables default outer join fetching.  **e.g.** recommended values between 0 and 3 |
| hibernate.default\_batch\_fetch\_size | Sets a default size for Hibernate batch fetching of associations.  **e.g.** recommended values 4, 8, 16 |
| hibernate.default\_entity\_mode | Sets a default mode for entity representation for all sessions opened from this SessionFactory  dynamic-map, dom4j, pojo |
| hibernate.order\_updates | Forces Hibernate to order SQL updates by the primary key value of the items being updated. This will result in fewer transaction deadlocks in highly concurrent systems.  **e.g.** true | false |
| hibernate.generate\_statistics | If enabled, Hibernate will collect statistics useful for performance tuning.  **e.g.** true | false |
| hibernate.use\_identifier\_rollback | If enabled, generated identifier properties will be reset to default values when objects are deleted.  **e.g.** true | false |
| hibernate.use\_sql\_comments | If turned on, Hibernate will generate comments inside the SQL, for easier debugging, defaults to false.  **e.g.** true | false |

**Hibernate JDBC and Connection Properties**

| **Property name** | **Purpose** |
| --- | --- |
| hibernate.jdbc.fetch\_size | A non-zero value determines the JDBC fetch size (calls Statement.setFetchSize()). |
| hibernate.jdbc.batch\_size | A non-zero value enables use of JDBC2 batch updates by Hibernate.  **e.g.** recommended values between 5 and 30 |
| hibernate.jdbc.batch\_versioned\_data | Set this property to true if your JDBC driver returns correct row counts from executeBatch(). Iit is usually safe to turn this option on. Hibernate will then use batched DML for automatically versioned data. Defaults to false.  **e.g.** true | false |
| hibernate.jdbc.factory\_class | Select a custom org.hibernate.jdbc.Batcher. Most applications will not need this configuration property.  **e.g.** classname.of.BatcherFactory |
| hibernate.jdbc.use\_scrollable\_resultset | Enables use of JDBC2 scrollable resultsets by Hibernate. This property is only necessary when using user-supplied JDBC connections. Hibernate uses connection metadata otherwise.  **e.g.** true | false |
| hibernate.jdbc.use\_streams\_for\_binary | Use streams when writing/reading binary or serializable types to/from JDBC. *\*system-level property\**  **e.g.** true | false |
| hibernate.jdbc.use\_get\_generated\_keys | Enables use of JDBC3 PreparedStatement.getGeneratedKeys() to retrieve natively generated keys after insert. Requires JDBC3+ driver and JRE1.4+, set to false if your driver has problems with the Hibernate identifier generators. By default, it tries to determine the driver capabilities using connection metadata.  **e.g.** true|false |
| hibernate.connection.provider\_class | The classname of a custom org.hibernate.connection.ConnectionProvider which provides JDBC connections to Hibernate.  **e.g.** classname.of.ConnectionProvider |
| hibernate.connection.isolation | Sets the JDBC transaction isolation level. Check java.sql.Connection for meaningful values, but note that most databases do not support all isolation levels and some define additional, non-standard isolations.  **e.g.** 1, 2, 4, 8 |
| hibernate.connection.autocommit | Enables autocommit for JDBC pooled connections (it is not recommended).  **e.g.** true | false |
| hibernate.connection.release\_mode | Specifies when Hibernate should release JDBC connections. By default, a JDBC connection is held until the session is explicitly closed or disconnected. For an application server JTA datasource, use after\_statement to aggressively release connections after every JDBC call. For a non-JTA connection, it often makes sense to release the connection at the end of each transaction, by using after\_transaction. auto will choose after\_statement for the JTA and CMT transaction strategies and after\_transaction for the JDBC transaction strategy.  **e.g.** auto (default) | on\_close | after\_transaction | after\_statement  This setting only affects Sessions returned from SessionFactory.openSession. For Sessions obtained through SessionFactory.getCurrentSession, the CurrentSessionContext implementation configured for use controls the connection release mode for those Sessions. See [Section 2.5, “Contextual sessions”](https://docs.jboss.org/hibernate/orm/3.3/reference/en/html/architecture.html#architecture-current-session) |
| hibernate.connection.*<propertyName>* | Pass the JDBC property *<propertyName>* to DriverManager.getConnection(). |
| hibernate.jndi.*<propertyName>* | Pass the property *<propertyName>* to the JNDI InitialContextFactory. |

**Hibernate Cache Properties**

| **Property name** | **Purpose** |
| --- | --- |
| hibernate.cache.provider\_class | The classname of a custom CacheProvider.  **e.g.** classname.of.CacheProvider |
| hibernate.cache.use\_minimal\_puts | Optimizes second-level cache operation to minimize writes, at the cost of more frequent reads. This setting is most useful for clustered caches and, in Hibernate3, is enabled by default for clustered cache implementations.  **e.g.** true|false |
| hibernate.cache.use\_query\_cache | Enables the query cache. Individual queries still have to be set cachable.  **e.g.** true|false |
| hibernate.cache.use\_second\_level\_cache | Can be used to completely disable the second level cache, which is enabled by default for classes which specify a <cache> mapping.  **e.g.** true|false |
| hibernate.cache.query\_cache\_factory | The classname of a custom QueryCache interface, defaults to the built-in StandardQueryCache.  **e.g.** classname.of.QueryCache |
| hibernate.cache.region\_prefix | A prefix to use for second-level cache region names.  **e.g.** prefix |
| hibernate.cache.use\_structured\_entries | Forces Hibernate to store data in the second-level cache in a more human-friendly format.  **e.g.** true|false |

**Hibernate Transaction Properties**

| **Property name** | **Purpose** |
| --- | --- |
| hibernate.transaction.factory\_class | The classname of a TransactionFactory to use with Hibernate Transaction API (defaults to JDBCTransactionFactory).  **e.g.** classname.of.TransactionFactory |
| jta.UserTransaction | A JNDI name used by JTATransactionFactory to obtain the JTA UserTransaction from the application server.  **e.g.** jndi/composite/name |
| hibernate.transaction.manager\_lookup\_class | The classname of a TransactionManagerLookup. It is required when JVM-level caching is enabled or when using hilo generator in a JTA environment.  **e.g.** classname.of.TransactionManagerLookup |
| hibernate.transaction.flush\_before\_completion | If enabled, the session will be automatically flushed during the before completion phase of the transaction. Built-in and automatic session context management is preferred, see [Section 2.5, “Contextual sessions”](https://docs.jboss.org/hibernate/orm/3.3/reference/en/html/architecture.html#architecture-current-session).  **e.g.** true | false |
| hibernate.transaction.auto\_close\_session | If enabled, the session will be automatically closed during the after completion phase of the transaction. Built-in and automatic session context management is preferred, see [Section 2.5, “Contextual sessions”](https://docs.jboss.org/hibernate/orm/3.3/reference/en/html/architecture.html#architecture-current-session).  **e.g.** true | false |

**Miscellaneous Properties**

| **Property name** | **Purpose** |
| --- | --- |
| hibernate.current\_session\_context\_class | Supply a custom strategy for the scoping of the "current" Session. See [Section 2.5, “Contextual sessions”](https://docs.jboss.org/hibernate/orm/3.3/reference/en/html/architecture.html#architecture-current-session) for more information about the built-in strategies.  **e.g.** jta | thread | managed | custom.Class |
| hibernate.query.factory\_class | Chooses the HQL parser implementation.  **e.g.** org.hibernate.hql.ast.ASTQueryTranslatorFactory or org.hibernate.hql.classic.ClassicQueryTranslatorFactory |
| hibernate.query.substitutions | Is used to map from tokens in Hibernate queries to SQL tokens (tokens might be function or literal names, for example).  **e.g.** hqlLiteral=SQL\_LITERAL, hqlFunction=SQLFUNC |
| hibernate.hbm2ddl.auto | Automatically validates or exports schema DDL to the database when the SessionFactory is created. With create-drop, the database schema will be dropped when the SessionFactory is closed explicitly.  **e.g.** validate | update | create | create-drop |
| hibernate.cglib.use\_reflection\_optimizer | Enables the use of CGLIB instead of runtime reflection (System-level property). Reflection can sometimes be useful when troubleshooting. Hibernate always requires CGLIB even if you turn off the optimizer. You cannot set this property in hibernate.cfg.xml.  **e.g.** true | false |

**SQL Dialects**

Always set the hibernate.dialect property to the correct org.hibernate.dialect.Dialect subclass for your database. If you specify a dialect, Hibernate will use sensible defaults for some of the other properties listed above. This means that you will not have to specify them manually.

**Hibernate SQL Dialects (hibernate.dialect)**

| **RDBMS** | **Dialect** |
| --- | --- |
| DB2 | org.hibernate.dialect.DB2Dialect |
| DB2 AS/400 | org.hibernate.dialect.DB2400Dialect |
| DB2 OS390 | org.hibernate.dialect.DB2390Dialect |
| PostgreSQL | org.hibernate.dialect.PostgreSQLDialect |
| MySQL | org.hibernate.dialect.MySQLDialect |
| MySQL with InnoDB | org.hibernate.dialect.MySQLInnoDBDialect |
| MySQL with MyISAM | org.hibernate.dialect.MySQLMyISAMDialect |
| Oracle (any version) | org.hibernate.dialect.OracleDialect |
| Oracle 9i | org.hibernate.dialect.Oracle9iDialect |
| Oracle 10g | org.hibernate.dialect.Oracle10gDialect |
| Sybase | org.hibernate.dialect.SybaseDialect |
| Sybase Anywhere | org.hibernate.dialect.SybaseAnywhereDialect |
| Microsoft SQL Server | org.hibernate.dialect.SQLServerDialect |
| SAP DB | org.hibernate.dialect.SAPDBDialect |
| Informix | org.hibernate.dialect.InformixDialect |
| HypersonicSQL | org.hibernate.dialect.HSQLDialect |
| Ingres | org.hibernate.dialect.IngresDialect |
| Progress | org.hibernate.dialect.ProgressDialect |
| Mckoi SQL | org.hibernate.dialect.MckoiDialect |
| Interbase | org.hibernate.dialect.InterbaseDialect |
| Pointbase | org.hibernate.dialect.PointbaseDialect |
| FrontBase | org.hibernate.dialect.FrontbaseDialect |
| Firebird | org.hibernate.dialect.FirebirdDialect |