34. List of all available configuration properties34.1. Hibernate’s core ORM functionality hibernate.allow\_refresh\_detached\_entity When enabled, allows calls to EntityManager.refresh(Object) and Session.refresh(Object) on a detached entity instance. Values are true, which allows refreshing a detached instance and false, which does not. When refreshing is disallowed, an IllegalArgumentException is thrown. The default behavior is to allow refreshing a detached instance unless Hibernate is bootstrapped via JPA. Since: 5.2 See Also: Constant Field Values hibernate.allow\_update\_outside\_transaction When enabled, allows update operations outside a transaction. Since version 5.2 Hibernate conforms with the JPA specification and disallows flushing any update outside a transaction. Values are true, which allows flushing outside a transaction, and false, which does not. The default behavior is to disallow update operations outside a transaction. Since: 5.2 See Also: SessionFactoryBuilder.allowOutOfTransactionUpdateOperations(boolean), Constant Field Values hibernate.archive.autodetection Identifies a comma-separated list of values indicating the types of things we should auto-detect during scanning. Allowable values include: "class" specifies that .class files are discovered as managed classes "hbm" specifies that hbm.xml files are discovered as mapping files See Also: MetadataBuilder.applyScanOptions(org.hibernate.boot.archive.scan.spi.ScanOptions), Constant Field Values hibernate.archive.interpreter Specifies an ArchiveDescriptorFactory to use in the scanning process, either: an instance of ArchiveDescriptorFactory, a Class representing a class that implements ArchiveDescriptorFactory, or the name of a class that implements ArchiveDescriptorFactory. See information on Scanner about expected constructor forms. See Also: SCANNER, Scanner, AbstractScannerImpl, MetadataBuilder.applyArchiveDescriptorFactory(org.hibernate.boot.archive.spi.ArchiveDescriptorFactory), Constant Field Values hibernate.archive.scanner Specifies an implementation of Scanner, either: an instance of Scanner, a Class representing a class that implements Scanner the name of a class that implements Scanner. See Also: MetadataBuilder.applyScanner(org.hibernate.boot.archive.scan.spi.Scanner), Constant Field Values hibernate.auto\_quote\_keyword Specifies whether to automatically quote any names that are deemed SQL keywords. Auto-quoting of SQL keywords is disabled by default. Since: 5.0 See Also: Constant Field Values hibernate.batch\_fetch\_style Deprecated. An appropriate batch-fetch style is selected automatically Specifies the BatchFetchStyle to use, either the name of a {code BatchFetchStyle} instance, or an instance of BatchFetchStyle. See Also: Constant Field Values hibernate.bytecode.provider Selects a bytecode enhancement library. At present only bytebuddy is supported, bytebuddy being the default since version 5.3. See Also: Constant Field Values hibernate.bytecode.use\_reflection\_optimizer Deprecated, for removal: This API element is subject to removal in a future version. Will be removed without replacement. See HHH-15631 When enabled, specifies that property access should be optimized via the use of generated bytecode. See Also: Constant Field Values hibernate.c3p0 A setting prefix used to indicate settings that target the hibernate-c3p0 integration See Also: Constant Field Values hibernate.c3p0.acquire\_increment Number of connections acquired when pool is exhausted See Also: Constant Field Values hibernate.c3p0.idle\_test\_period Idle time before a C3P0 pooled connection is validated See Also: Constant Field Values hibernate.c3p0.max\_size Maximum size of C3P0 connection pool See Also: Constant Field Values hibernate.c3p0.max\_statements Maximum size of C3P0 statement cache See Also: Constant Field Values hibernate.c3p0.min\_size Minimum size of C3P0 connection pool See Also: Constant Field Values hibernate.c3p0.timeout Maximum idle time for C3P0 connection pool See Also: Constant Field Values hibernate.cache.auto\_evict\_collection\_cache Enables the automatic eviction of a bidirectional association’s collection cache when an element in the ManyToOne collection is added, updated, or removed without properly managing the change on the OneToMany side. See Also: SessionFactoryBuilder.applyAutomaticEvictionOfCollectionCaches(boolean), Constant Field Values hibernate.cache.default\_cache\_concurrency\_strategy Specifies the CacheConcurrencyStrategy to use by default when an entity is marked @Cacheable, but no concurrency strategy is explicitly specified via the Cache annotation. An explicit strategy may be specified using @Cache(usage=…). See Also: MetadataBuilder.applyAccessType(org.hibernate.cache.spi.access.AccessType), Constant Field Values hibernate.cache.keys\_factory Deprecated. this is only honored for hibernate-infinispan Specifies the CacheKeysFactory to use, either: an instance of CacheKeysFactory, a Class implementing CacheKeysFactory, the name of a class implementing CacheKeysFactory, "default" as a short name for DefaultCacheKeysFactory, or "simple" as a short name for SimpleCacheKeysFactory. Since: 5.2 See Also: Constant Field Values hibernate.cache.query\_cache\_factory Specifies the TimestampsCacheFactory to use. See Also: SessionFactoryBuilder.applyTimestampsCacheFactory(TimestampsCacheFactory), Constant Field Values hibernate.cache.region.factory\_class The RegionFactory implementation, either: an instance of RegionFactory, a Class implementing RegionFactory, or he name of a class implementing RegionFactory. Defaults to NoCachingRegionFactory, so that caching is disabled. See Also: USE\_SECOND\_LEVEL\_CACHE, Constant Field Values hibernate.cache.region\_prefix The CacheProvider region name prefix See Also: SessionFactoryBuilder.applyCacheRegionPrefix(String), Constant Field Values hibernate.cache.use\_minimal\_puts Optimize interaction with the second-level cache to minimize writes, at the cost of an additional read before each write. This setting is useful if writes to the cache are much more expensive than reads from the cache, for example, if the cache is a distributed cache. It’s not usually necessary to set this explicitly because, by default, it’s set to a sensible value by the second-level cache implementation. See Also: SessionFactoryBuilder.applyMinimalPutsForCaching(boolean), Constant Field Values hibernate.cache.use\_query\_cache Enable the query cache (disabled by default). See Also: SessionFactoryBuilder.applyQueryCacheSupport(boolean), Constant Field Values hibernate.cache.use\_reference\_entries Enable direct storage of entity references into the second level cache when applicable. This is appropriate only for immutable entities. By default, entities are always stored in a "disassembled" form, that is, as a tuple of attribute values. See Also: SessionFactoryBuilder.applyDirectReferenceCaching(boolean), Constant Field Values hibernate.cache.use\_second\_level\_cache When enabled, specifies that the second-level cache may be used. By default, if the configured RegionFactory is not the NoCachingRegionFactory, then the second-level cache is enabled. Otherwise, the second-level cache is disabled. See Also: CACHE\_REGION\_FACTORY, SessionFactoryBuilder.applySecondLevelCacheSupport(boolean), Constant Field Values hibernate.cache.use\_structured\_entries Enables the use of structured second-level cache entries. This makes the cache entries human-readable, but carries a performance cost. See Also: SessionFactoryBuilder.applyStructuredCacheEntries(boolean), Constant Field Values hibernate.cdi.extensions Controls whether Hibernate can try to create beans other than converters and listeners using CDI. Only meaningful when a CDI container is used. By default, Hibernate will only attempt to create converter and listener beans using CDI. Since: 6.2 See Also: Constant Field Values hibernate.cfg\_xml\_file See Also: Constant Field Values hibernate.check\_nullability Enable nullability checking, raises an exception if an attribute marked as not null is null at runtime. Defaults to disabled if Bean Validation is present in the classpath and annotations are used, or enabled otherwise. See Also: SessionFactoryBuilder.applyNullabilityChecking(boolean), Constant Field Values hibernate.classLoader.tccl\_lookup\_precedence Specifies how the thread context class loader must be used for class lookup. See Also: TcclLookupPrecedence, Constant Field Values hibernate.classLoaders Specifies a collection of the ClassLoader instances Hibernate should use for classloading and resource loading. Since: 5.0 See Also: Constant Field Values hibernate.classcache Entity cache configuration properties follow the pattern hibernate.classcache.packagename.ClassName usage[, region] where usage is the cache strategy used and region the cache region name See Also: Constant Field Values hibernate.collectioncache Collection cache configuration properties follow the pattern hibernate.collectioncache.packagename.ClassName.role usage[, region] where usage is the cache strategy used and region the cache region name See Also: Constant Field Values hibernate.column\_ordering\_strategy Used to specify the ColumnOrderingStrategy class to use. The following shortcut names are defined for this setting: "default" is an abbreviations for ColumnOrderingStrategyStandard "legacy" is an abbreviation for ColumnOrderingStrategyLegacy By default, the ColumnOrderingStrategy registered under the key "default" is used. If no strategy is explicitly registered under that key, ColumnOrderingStrategyStandard is used. Since: 6.2 See Also: MetadataBuilder.applyColumnOrderingStrategy(org.hibernate.boot.model.relational.ColumnOrderingStrategy), Constant Field Values hibernate.connection A prefix for properties specifying arbitrary JDBC connection properties. These properties are simply passed along to the provider when creating a connection. See Also: Constant Field Values hibernate.connection.autocommit Controls the autocommit mode of JDBC connections obtained from any ConnectionProvider implementation which respects this setting, which the built-in implementations do, except for DatasourceConnectionProviderImpl. See Also: Constant Field Values hibernate.connection.datasource Specifies a DataSource, either: an instance of DataSource, or a JNDI name under which to obtain the DataSource. For JNDI names, see also JNDI\_CLASS, JNDI\_URL, JNDI\_PREFIX, etc. See Also: DataSource, JAKARTA\_JTA\_DATASOURCE, JAKARTA\_NON\_JTA\_DATASOURCE, Constant Field Values hibernate.connection.driver\_class Specifies the JDBC driver class. See Also: Driver, JAKARTA\_JDBC\_DRIVER, Constant Field Values hibernate.connection.handling\_mode Specifies how Hibernate should manage JDBC connections in terms of acquisition and release, either: an instance of the enumeration PhysicalConnectionHandlingMode, or the name of one of its instances. The default is DELAYED\_ACQUISITION\_AND\_RELEASE\_AFTER\_TRANSACTION. Since: 5.2 See Also: PhysicalConnectionHandlingMode, SessionFactoryBuilder.applyConnectionHandlingMode(PhysicalConnectionHandlingMode), Constant Field Values hibernate.connection.isolation Specified the JDBC transaction isolation level. See Also: Constant Field Values hibernate.connection.password Specifies password to use when connecting via JDBC. See Also: USER, JAKARTA\_JDBC\_USER, Constant Field Values hibernate.connection.pool\_size Specifies the maximum number of inactive connections for the built-in connection pool. See Also: Constant Field Values hibernate.connection.provider\_class Specifies a ConnectionProvider to use for obtaining JDBC connections, either: an instance of ConnectionProvider, a Class representing a class that implements ConnectionProvider, or the name of a class that implements ConnectionProvider. The term "class" appears in the setting name due to legacy reasons; however it can accept instances. See Also: Constant Field Values hibernate.connection.provider\_disables\_autocommit Allows a user to tell Hibernate that the connections we obtain from the configured ConnectionProvider will already have autocommit disabled when we acquire them from the provider. When we obtain connections with autocommit already disabled, we may circumvent some operations in the interest of performance. By default, Hibernate calls Connection.setAutoCommit(boolean) on newly-obtained connections. Since: 5.2.10 See Also: SessionFactoryBuilder.applyConnectionProviderDisablesAutoCommit(boolean), Constant Field Values hibernate.connection.url Specifies the JDBC connection URL. See Also: JAKARTA\_JDBC\_URL, Constant Field Values hibernate.connection.username Specifies the database user to use when connecting via JDBC. Depending on the configured ConnectionProvider, the specified username might be used to: create a JDBC connection using DriverManager.getConnection(String,java.util.Properties) or Driver.connect(String,java.util.Properties), or obtain a JDBC connection from a datasource, using DataSource.getConnection(String, String). See Also: PASS, JAKARTA\_JDBC\_PASSWORD, Constant Field Values hibernate.create\_empty\_composites.enabled Deprecated. It makes no sense at all to enable this at the global level for a persistence unit. If anything, it could be a setting specific to a given embeddable class. But, four years after the introduction of this feature, it’s still marked experimental and has multiple known unresolved bugs. It’s therefore time for those who advocated for this feature to accept defeat. Enable instantiation of composite/embedded objects when all attribute values are null. The default (and historical) behavior is that a null reference will be used to represent the composite value when all of its attributes are null. Since: 5.1 See Also: Constant Field Values "API Note:" This is an experimental feature that has known issues. It should not be used in production until it is stabilized. See Hibernate JIRA issue HHH-11936 for details. hibernate.criteria.copy\_tree When enabled, specifies that queries created through EntityManager.createQuery(CriteriaQuery), EntityManager.createQuery(CriteriaUpdate) or EntityManager.createQuery(CriteriaDelete) must create a copy of the passed object such that the resulting Query is not affected by any mutations to the original criteria query. If disabled, it is assumed that users do not mutate the criteria query afterwards and due to that, no copy will be created, which will improve performance. When bootstrapping Hibernate through the native bootstrap APIs this setting is disabled i.e. no copies are created to not hurt performance. When bootstrapping Hibernate through the JPA SPI this setting is enabled. When enabled, criteria query objects are copied, as required by the Jakarta Persistence specification. Since: 6.0 See Also: Constant Field Values hibernate.criteria.value\_handling\_mode By default, criteria queries use bind parameters for any value passed via the JPA Criteria API. The "bind" mode uses bind variables for any literal value. The "inline" mode inlines values as SQL literals. The default value is ValueHandlingMode.BIND. Since: 6.0.0 See Also: ValueHandlingMode, Constant Field Values hibernate.current\_session\_context\_class Specifies a CurrentSessionContext for scoping the current session, either: jta, thread, or managed, or the name of a class implementing org.hibernate.context.spi.CurrentSessionContext. If this property is not set, but JTA support is enabled, then JTASessionContext is used by default. See Also: SessionFactory.getCurrentSession(), CurrentSessionContext, Constant Field Values hibernate.default\_batch\_fetch\_size Specifies the default batch size for batch fetching. See Also: BatchSize, SessionFactoryBuilder.applyDefaultBatchFetchSize(int), Constant Field Values hibernate.default\_catalog A default database catalog name to use for unqualified table names See Also: MetadataBuilder.applyImplicitCatalogName(java.lang.String), Constant Field Values hibernate.default\_schema A default database schema (owner) name to use for unqualified table names See Also: MetadataBuilder.applyImplicitSchemaName(java.lang.String), Constant Field Values hibernate.delay\_cdi\_access Used in conjunction with "hibernate.resource.beans.container" when CDI is used. By default, to be JPA spec compliant, Hibernate should access the CDI BeanManager while bootstrapping the SessionFactory. In some cases however this can lead to a chicken/egg situation where the JPA provider immediately accesses the BeanManager when managed beans are awaiting JPA PU injection. This setting tells Hibernate to delay accessing until first use. This setting has the decided downside that bean config problems will not be done at deployment time, but will instead manifest at runtime. For this reason, the preferred means for supplying a CDI BeanManager is to provide an implementation of ExtendedBeanManager which gives Hibernate a callback when the BeanManager is ready for use. Since: 5.0.8 See Also: Constant Field Values hibernate.dialect Specifies the Hibernate SQL dialect, either an instance of Dialect, a Class representing a class that extends Dialect, or the name of a class that extends Dialect. By default, Hibernate will attempt to automatically determine the dialect from the JDBC URL and JDBC metadata, so this setting is not usually necessary. See Also: Dialect, Constant Field Values hibernate.dialect.native\_param\_markers Controls whether to use JDBC markers (?) or dialect native markers for parameters within preparable SQL statements. Since: 6.2 See Also: ParameterMarkerStrategy, Dialect.getNativeParameterMarkerStrategy(), Constant Field Values "Implementation Note:" False by default, indicating standard JDBC parameter markers (?) are used. Set to true to use the Dialect’s native markers, if any. For Dialects without native markers, the standard JDBC strategy is used. hibernate.dialect.storage\_engine Specifies the default storage engine for a relational databases that supports multiple storage engines. This property must be set either as an Environment variable or JVM System Property, since the Dialect is instantiated before Hibernate property resolution. Since: 5.2.9 See Also: Constant Field Values hibernate.dialect\_resolvers Specifies additional DialectResolver implementations to register with the standard DialectFactory. See Also: Constant Field Values hibernate.discard\_pc\_on\_close When enabled, specifies that the persistent context should be discarded when either SharedSessionContract.close() or EntityManager.close() is called. By default, the persistent context is not discarded, as per the JPA specification. See Also: Constant Field Values hibernate.discriminator.force\_in\_select See Also: MetadataBuilder.enableImplicitForcingOfDiscriminatorsInSelect(boolean), Constant Field Values hibernate.discriminator.ignore\_explicit\_for\_joined The legacy behavior of Hibernate is to not use discriminators for joined inheritance (Hibernate does not need the discriminator). However, some JPA providers do need the discriminator for handling joined inheritance, so in the interest of portability this capability has been added to Hibernate. Existing applications rely (implicitly or explicitly) on Hibernate ignoring any DiscriminatorColumn declarations on joined inheritance hierarchies. This setting allows these applications to maintain the legacy behavior of @DiscriminatorColumn annotations being ignored when paired with joined inheritance. See Hibernate Jira issue HHH-6911 for additional background info. This setting defaults to false, meaning that explicit discriminator columns are never ignored. See Also: MetadataBuilder.enableExplicitDiscriminatorsForJoinedSubclassSupport(boolean), IMPLICIT\_DISCRIMINATOR\_COLUMNS\_FOR\_JOINED\_SUBCLASS, Constant Field Values hibernate.discriminator.implicit\_for\_joined The legacy behavior of Hibernate is to not use discriminators for joined inheritance (Hibernate does not need the discriminator.). However, some JPA providers do need the discriminator for handling joined inheritance, so in the interest of portability this capability has been added to Hibernate. However, we want to make sure that legacy applications continue to work as well. Which puts us in a bind in terms of how to handle "implicit" discriminator mappings. The solution is to assume that the absence of discriminator metadata means to follow the legacy behavior unless this setting is enabled. With this setting enabled, Hibernate will interpret the absence of discriminator metadata as an indication to use the JPA defined defaults for these absent annotations. See Hibernate Jira issue HHH-6911 for additional background info. This setting defaults to false, meaning that implicit discriminator columns are never inferred to exist for joined inheritance hierarchies. See Also: MetadataBuilder.enableImplicitDiscriminatorsForJoinedSubclassSupport(boolean), IGNORE\_EXPLICIT\_DISCRIMINATOR\_COLUMNS\_FOR\_JOINED\_SUBCLASS, Constant Field Values hibernate.enable\_lazy\_load\_no\_trans Allows a detached proxy or lazy collection to be fetched even when not associated with an open persistence context, by creating a temporary persistence context when the proxy or collection is accessed. This behavior is not recommended, since it can easily break transaction isolation or lead to data aliasing. It is therefore disabled by default. See Also: SessionFactoryBuilder.applyLazyInitializationOutsideTransaction(boolean), Constant Field Values hibernate.enhancer.enableAssociationManagement Enable association management feature in runtime bytecode enhancement See Also: Constant Field Values hibernate.enhancer.enableDirtyTracking Deprecated, for removal: This API element is subject to removal in a future version. Will be removed without replacement. See HHH-15641 Enable dirty tracking feature in runtime bytecode enhancement See Also: Constant Field Values hibernate.enhancer.enableLazyInitialization Deprecated, for removal: This API element is subject to removal in a future version. Will be removed without replacement. See HHH-15641 Enable lazy loading feature in runtime bytecode enhancement See Also: Constant Field Values hibernate.entity\_dirtiness\_strategy Setting to identify a CustomEntityDirtinessStrategy to use. May specify either a class name or an instance. See Also: SessionFactoryBuilder.applyCustomEntityDirtinessStrategy(CustomEntityDirtinessStrategy), Constant Field Values hibernate.event.listener Event listener configuration properties follow the pattern hibernate.event.listener.eventType packageName.ClassName1, packageName.ClassName2 See Also: Constant Field Values hibernate.event.merge.entity\_copy\_observer Setting that specifies how Hibernate will respond when multiple representations of the same persistent entity ("entity copy") are detected while merging. The possible values are: disallow (the default): throws IllegalStateException if an entity copy is detected allow: performs the merge operation on each entity copy that is detected log: (provided for testing only) performs the merge operation on each entity copy that is detected and logs information about the entity copies. This setting requires DEBUG logging be enabled for EntityCopyAllowedLoggedObserver. Alternatively, the application may customize the behavior by providing an implementation of EntityCopyObserver and setting the property "hibernate.event.merge.entity\_copy\_observer" to the class name. When this property is set to allow or log, Hibernate will merge each entity copy detected while cascading the merge operation. In the process of merging each entity copy, Hibernate will cascade the merge operation from each entity copy to its associations with CascadeType.MERGE or CascadeType.ALL. The entity state resulting from merging an entity copy will be overwritten when another entity copy is merged. Since: 4.3 See Also: Constant Field Values hibernate.format\_sql Enables formatting of SQL logged to the console. See Also: Constant Field Values hibernate.generate\_statistics When enabled, specifies that statistics should be collected. See Also: SessionFactoryBuilder.applyStatisticsSupport(boolean), Constant Field Values hibernate.globally\_quoted\_identifiers When enabled, all database identifiers are quoted. See Also: Constant Field Values hibernate.globally\_quoted\_identifiers\_skip\_column\_definitions Assuming GLOBALLY\_QUOTED\_IDENTIFIERS, this allows global quoting to skip column definitions defined by Column, JoinColumn, etc. JPA states that column definitions are subject to global quoting, so by default this setting is false for JPA compliance. Set to true to avoid explicit column names being quoted due to global quoting (they will still be quoted if explicitly quoted in the annotation or XML). See Also: Constant Field Values hibernate.hbm2ddl.auto Setting to perform SchemaManagementTool actions automatically as part of the SessionFactory lifecycle. Valid options are enumerated by Action. Interpreted in combination with JAKARTA\_HBM2DDL\_DATABASE\_ACTION and JAKARTA\_HBM2DDL\_SCRIPTS\_ACTION. If no value is specified, the default is "none". See Also: Action, Constant Field Values hibernate.hbm2ddl.charset\_name The name of the charset used by the schema generation resource. By default, the JVM default charset is used. Since: 5.2.3 See Also: Constant Field Values hibernate.hbm2ddl.create\_namespaces Specifies whether to automatically create also the database schema/catalog. The default is false. Since: 5.0 See Also: Constant Field Values hibernate.hbm2ddl.default\_constraint\_mode Used with the ConstraintMode.PROVIDER\_DEFAULT strategy for foreign key mapping. Valid values are ConstraintMode.CONSTRAINT and ConstraintMode.NO\_CONSTRAINT. The default value is ConstraintMode.CONSTRAINT. Since: 5.4 See Also: Constant Field Values hibernate.hbm2ddl.delimiter Identifies the delimiter to use to separate schema management statements in script outputs. The default value is ;. See Also: Constant Field Values hibernate.hbm2ddl.extra\_physical\_table\_types Specifies a comma-separated list of extra table types, in addition to the default types "TABLE" and "VIEW", to recognize as physical tables when performing schema update, creation and validation. Since: 5.0 See Also: Constant Field Values hibernate.hbm2ddl.halt\_on\_error When enabled, specifies that the schema migration tool should halt on any error, terminating the bootstrap process. Since: 5.2.4 See Also: Constant Field Values hibernate.hbm2ddl.import\_files Specifies a comma-separated list of file names of scripts containing SQL DML statements that should be executed after schema export completes. The order of the scripts is significant, with the first script in the list being executed first. The scripts are only executed if the schema is created by Hibernate, that is, if "hibernate.hbm2ddl.auto" is set to create or create-drop. The default value is /import.sql. The JPA-standard setting JAKARTA\_HBM2DDL\_CREATE\_SCRIPT\_SOURCE is now preferred. See Also: Constant Field Values hibernate.hbm2ddl.import\_files\_sql\_extractor The SqlScriptCommandExtractor implementation to use for parsing source/import files specified by JAKARTA\_HBM2DDL\_CREATE\_SCRIPT\_SOURCE, JAKARTA\_HBM2DDL\_DROP\_SCRIPT\_SOURCE or HBM2DDL\_IMPORT\_FILES. Either: an instance of SqlScriptCommandExtractor, a Class object representing a class that implements SqlScriptCommandExtractor, or the name of a class that implements SqlScriptCommandExtractor. The correct extractor to use depends on the format of the SQL script: if the script has one complete SQL statement per line, use SingleLineSqlScriptExtractor, or if a script contains statements spread over multiple lines, use MultiLineSqlScriptExtractor. The default value is org.hibernate.tool.schema.internal.script.SingleLineSqlScriptExtractor. See Also: SingleLineSqlScriptExtractor, MultiLineSqlScriptExtractor, Constant Field Values hibernate.hbm2ddl.jdbc\_metadata\_extraction\_strategy Setting to choose the strategy used to access the JDBC Metadata. Valid options are defined by JdbcMetadaAccessStrategy. JdbcMetadaAccessStrategy.GROUPED is the default. See Also: JdbcMetadaAccessStrategy, Constant Field Values hibernate.hbm2ddl.schema-generation.script.append For cases where the "javax.persistence.schema-generation.scripts.action" value indicates that schema commands should be written to DDL script file, specifies if schema commands should be appended to the end of the file rather than written at the beginning of the file. Values are: true for appending schema commands to the end of the file, false for writing schema commands at the beginning. The default value is true See Also: Constant Field Values hibernate.hbm2ddl.schema\_filter\_provider Used to specify the SchemaFilterProvider to be used by create, drop, migrate and validate operations on the database schema. A SchemaFilterProvider provides filters that can be used to limit the scope of these operations to specific namespaces, tables and sequences. All objects are included by default. Since: 5.1 See Also: Constant Field Values hibernate.hbm\_xml\_files See Also: Constant Field Values hibernate.highlight\_sql Enables highlighting of SQL logged to the console using ANSI escape codes. See Also: Constant Field Values hibernate.hql.bulk\_id\_strategy.global\_temporary.create\_tables Allows creation of global temporary tables at application startup to be disabled. By default, table creation is enabled. See Also: Constant Field Values hibernate.hql.bulk\_id\_strategy.global\_temporary.drop\_tables Allows dropping of global temporary tables at application shutdown to be disabled. By default, table dropping is enabled. See Also: Constant Field Values hibernate.hql.bulk\_id\_strategy.local\_temporary.drop\_tables Allows dropping of local temporary tables at transaction commit to be enabled. By default, table dropping is disabled, and the database will drop the temporary tables automatically. See Also: Constant Field Values hibernate.hql.bulk\_id\_strategy.persistent.create\_tables Allows creation of persistent temporary tables at application startup to be disabled. By default, table creation is enabled. See Also: Constant Field Values hibernate.hql.bulk\_id\_strategy.persistent.drop\_tables Allows dropping of persistent temporary tables at application shutdown to be disabled. By default, table dropping is enabled. See Also: Constant Field Values hibernate.id.db\_structure\_naming\_strategy An implicit naming strategy for database structures (tables, sequences) related to identifier generators. Resolution uses the StrategySelector service and accepts any of the forms discussed on StrategySelector.resolveDefaultableStrategy(Class, Object, java.util.concurrent.Callable). The recognized short names being: "single" "legacy" "standard" See Also: ImplicitDatabaseObjectNamingStrategy, Constant Field Values hibernate.id.generator.stored\_last\_used Determines if the identifier value stored in the database table backing a table generator is the last value returned by the identifier generator, or the next value to be returned. By default, the value stored in the database table is the last generated value. Since: 5.3 See Also: Constant Field Values hibernate.id.optimizer.pooled.preferred When a generator specifies an increment-size and an optimizer was not explicitly specified, which of the "pooled" optimizers should be preferred? Can specify an optimizer short name or the name of a class which implements Optimizer. See Also: Constant Field Values hibernate.id.sequence.increment\_size\_mismatch\_strategy This setting defines the SequenceMismatchStrategy used when Hibernate detects a mismatch between a sequence configuration in an entity mapping and its database sequence object counterpart. Possible values are SequenceMismatchStrategy.EXCEPTION, SequenceMismatchStrategy.LOG, SequenceMismatchStrategy.FIX and SequenceMismatchStrategy.NONE. The default value is SequenceMismatchStrategy.EXCEPTION, meaning that an exception is thrown when such a conflict is detected. Since: 5.4 See Also: Constant Field Values hibernate.identifier\_generator\_strategy\_provider Deprecated. use GenerationTypeStrategyRegistration instead Specifies a class which implements IdentifierGeneratorStrategyProvider, and has a constructor with no parameters. See Also: Constant Field Values hibernate.implicit\_naming\_strategy Used to specify the ImplicitNamingStrategy class to use. The following shortcut names are defined for this setting: "default" and "jpa" are an abbreviations for ImplicitNamingStrategyJpaCompliantImpl "legacy-jpa" is an abbreviation for ImplicitNamingStrategyLegacyJpaImpl "legacy-hbm" is an abbreviation for ImplicitNamingStrategyLegacyHbmImpl "component-path" is an abbreviation for ImplicitNamingStrategyComponentPathImpl By default, the ImplicitNamingStrategy registered under the key "default" is used. If no strategy is explicitly registered under that key, ImplicitNamingStrategyJpaCompliantImpl is used. Since: 5.0 See Also: MetadataBuilder.applyImplicitNamingStrategy(org.hibernate.boot.model.naming.ImplicitNamingStrategy), Constant Field Values hibernate.integrator\_provider Names a IntegratorProvider See Also: Constant Field Values hibernate.jdbc.batch\_size Specifies the maximum JDBC batch size. A nonzero value enables batch updates. See Also: Statement.executeBatch(), SessionFactoryBuilder.applyJdbcBatchSize(int), Constant Field Values hibernate.jdbc.batch\_versioned\_data When enabled, specifies that versioned data should be included in batching. See Also: SessionFactoryBuilder.applyJdbcBatchingForVersionedEntities(boolean), Constant Field Values hibernate.jdbc.factory\_class Specifies a custom BatchBuilder. See Also: Constant Field Values hibernate.jdbc.fetch\_size Gives the JDBC driver a hint as to the number of rows that should be fetched from the database when more rows are needed. If 0, the JDBC driver’s default settings will be used. See Also: Statement.setFetchSize(int), SessionFactoryBuilder.applyJdbcFetchSize(int), ScrollableResults.setFetchSize(int), Constant Field Values hibernate.jdbc.lob.non\_contextual\_creation When enabled, specifies that Hibernate should not use contextual LOB creation. See Also: LobCreator, LobCreationContext, Constant Field Values hibernate.jdbc.log.warnings When enabled, specifies that JDBC statement warnings should be logged. The default is determined by Dialect.isJdbcLogWarningsEnabledByDefault(). Since: 5.1 See Also: Statement.getWarnings(), Constant Field Values hibernate.jdbc.time\_zone Specifies the time zone to use in the JDBC driver, which is supposed to match the database timezone. This is the timezone what will be passed to PreparedStatement.setTimestamp(int, java.sql.Timestamp, java.util.Calendar) PreparedStatement.setTime(int, java.sql.Time, java.util.Calendar), ResultSet.getTimestamp(int, Calendar), and ResultSet.getTime(int, Calendar) when binding parameters. The time zone may be given as: an instance of TimeZone, an instance of ZoneId, or a time zone ID string to be passed to ZoneId.of(String). By default, the JVM default time zone is assumed by the JDBC driver. Since: 5.2.3 See Also: Constant Field Values hibernate.jdbc.use\_get\_generated\_keys Specifies that generated primary keys may be retrieved using the JDBC 3 Statement.getGeneratedKeys() operation. Usually, performance will be improved if this behavior is enabled, assuming the JDBC driver supports getGeneratedKeys(). See Also: Statement.getGeneratedKeys(), SessionFactoryBuilder.applyGetGeneratedKeysSupport(boolean), Constant Field Values hibernate.jdbc.use\_scrollable\_resultset When enabled, specifies that JDBC scrollable ResultSet`s may be used. This property is only necessary when there is no `ConnectionProvider, that is, when the client is supplying JDBC connections. See Also: SessionFactoryBuilder.applyScrollableResultsSupport(boolean), Constant Field Values hibernate.jndi A prefix for properties specifying arbitrary JNDI InitialContext properties. These properties are simply passed along to the constructor InitialContext(java.util.Hashtable). See Also: Constant Field Values hibernate.jndi.class Specifies the JNDI InitialContext implementation class. See Also: Context.INITIAL\_CONTEXT\_FACTORY, Constant Field Values hibernate.jndi.url Specifies the JNDI provider/connection URL. See Also: Context.PROVIDER\_URL, Constant Field Values hibernate.jpa.compliance Specifies a default value for all JpaCompliance flags. Each individual flag may still be overridden by explicitly specifying its specific configuration property. Since: 6.0 See Also: JPA\_TRANSACTION\_COMPLIANCE, JPA\_QUERY\_COMPLIANCE, JPA\_LIST\_COMPLIANCE, JPA\_ORDER\_BY\_MAPPING\_COMPLIANCE, JPA\_CLOSED\_COMPLIANCE, JPA\_PROXY\_COMPLIANCE, JPA\_CACHING\_COMPLIANCE, JPA\_ID\_GENERATOR\_GLOBAL\_SCOPE\_COMPLIANCE, JPA\_LOAD\_BY\_ID\_COMPLIANCE, Constant Field Values hibernate.jpa.compliance.caching By default, Hibernate uses second-level cache invalidation for entities with secondary tables in order to avoid the possibility of inconsistent cached data in the case where different transactions simultaneously update different table rows corresponding to the same entity instance. The JPA TCK, for no good reason, requires that entities with secondary tables be immediately cached in the second-level cache rather than invalidated and re-cached on a subsequent read. Note that Hibernate’s default behavior here is safer and more careful than the behavior mandated by the TCK but YOLO. When enabled, this setting makes Hibernate pass the TCK. Since: 5.3 See Also: JpaCompliance.isJpaCacheComplianceEnabled(), AbstractEntityPersister.isCacheInvalidationRequired(), Constant Field Values hibernate.jpa.compliance.closed JPA specifies that an IllegalStateException must be thrown by EntityManager.close() and EntityManagerFactory.close() if the object has already been closed. By default, Hibernate treats any additional call to close() as a noop. When enabled, this setting forces Hibernate to throw an exception if close() is called on an instance that was already closed. Since: 5.3 See Also: JpaCompliance.isJpaClosedComplianceEnabled(), SessionFactoryBuilder.enableJpaClosedCompliance(boolean), Constant Field Values hibernate.jpa.compliance.global\_id\_generators Determines whether the scope of any identifier generator name specified via TableGenerator.name() or SequenceGenerator.name() is considered global to the persistence unit, or local to the entity in which identifier generator is defined. If enabled, the name will be considered globally scoped, and so the existence of two different generators with the same name will be considered a collision, and will result in an exception during bootstrap. Since: 5.2.17 See Also: JpaCompliance.isGlobalGeneratorScopeEnabled(), Constant Field Values hibernate.jpa.compliance.list Deprecated. Use DEFAULT\_LIST\_SEMANTICS instead. The specification actually leaves this behavior undefined, saying that portable applications should not rely on any specific behavior for a List with no @OrderColumn. Controls whether Hibernate should treat what it would usually consider a "bag", that is, a list with no index column, whose element order is not persistent, as a true list with an index column and a persistent element order. If enabled, Hibernate will recognize it as a list where the OrderColumn annotation is simply missing (and its defaults will apply). Since: 5.3 See Also: JpaCompliance.isJpaListComplianceEnabled(), SessionFactoryBuilder.enableJpaListCompliance(boolean), Constant Field Values hibernate.jpa.compliance.load\_by\_id Determines if an identifier value passed to EntityManager.find(java.lang.Class<T>, java.lang.Object) or EntityManager.getReference(java.lang.Class<T>, java.lang.Object) may be coerced to the identifier type declared by the entity. For example, an Integer argument might be widened to Long. By default, coercion is allowed. When enabled, coercion is disallowed, as required by the JPA specification. Since: 6.0 See Also: JpaCompliance.isLoadByIdComplianceEnabled(), Constant Field Values hibernate.jpa.compliance.orderby JPA specifies that items occurring in OrderBy lists must be references to entity attributes, whereas Hibernate, by default, allows more complex expressions. If enabled, an exception is thrown for items which are not entity attribute references. Since: 6.0 See Also: JpaCompliance.isJpaOrderByMappingComplianceEnabled(), SessionFactoryBuilder.enableJpaOrderByMappingCompliance(boolean), Constant Field Values hibernate.jpa.compliance.proxy The JPA specification insists that an EntityNotFoundException must be thrown whenever an uninitialized entity proxy with no corresponding row in the database is accessed. For most programs, this results in many completely unnecessary round trips to the database. Traditionally, Hibernate does not initialize an entity proxy when its identifier attribute is accessed, since the identifier value is already known and held in the proxy instance. This behavior saves the round trip to the database. When enabled, this setting forces Hibernate to initialize the entity proxy when its identifier is accessed. Clearly, this setting is not recommended. Since: 5.2.13 See Also: JpaCompliance.isJpaProxyComplianceEnabled(), Constant Field Values hibernate.jpa.compliance.query When enabled, specifies that every query must strictly follow the specified behavior of Query. The affects JPQL queries, criteria queries, and native SQL queries. This setting modifies the behavior of the JPQL query translator, and of the Query interface itself. In particular, it forces all methods of Query to throw the exception types defined by the JPA specification. If enabled, any deviations from the JPQL specification results in an exception. Therefore, this setting is not recommended, since it prohibits the use of many useful features of HQL. Since: 5.3 See Also: JpaCompliance.isJpaQueryComplianceEnabled(), SessionFactoryBuilder.enableJpaQueryCompliance(boolean), Constant Field Values hibernate.jpa.compliance.transaction When enabled, specifies that the Hibernate Transaction should behave according to the semantics defined by the JPA specification for an EntityTransaction. Since: 5.3 See Also: JpaCompliance.isJpaTransactionComplianceEnabled(), SessionFactoryBuilder.enableJpaTransactionCompliance(boolean), Constant Field Values hibernate.jpa.metamodel.population Setting that indicates whether to build the JPA types, either: enabled - Do the build disabled - Do not do the build ignoreUnsupported - Do the build, but ignore any non-JPA features that would otherwise result in a failure. See Also: Constant Field Values hibernate.jpa.static\_metamodel.population Setting that controls whether we seek out JPA "static metamodel" classes and populate them, either: enabled - Do the population disabled - Do not do the population skipUnsupported - Do the population, but ignore any non-JPA features that would otherwise result in the population failing. See Also: Constant Field Values hibernate.jpa\_callbacks.enabled Allows JPA callbacks (via PreUpdate and friends) to be completely disabled. Mostly useful to save some memory when they are not used. JPA callbacks are enabled by default. Set this property to false to disable them. Experimental and will likely be removed as soon as the memory overhead is resolved. Since: 5.4 See Also: CallbackType, Constant Field Values hibernate.jta.allowTransactionAccess When enabled, allows access to the Transaction even when using a JTA for transaction management. Values are true, which grants access, and false, which does not. The default behavior is to allow access unless Hibernate is bootstrapped via JPA. See Also: Constant Field Values hibernate.jta.cacheTransactionManager When enabled, indicates that it is safe to cache TransactionManager references. Since: 4.0 See Also: Constant Field Values hibernate.jta.cacheUserTransaction When enabled, indicates that it is safe to cache UserTransaction references. Since: 4.0 See Also: Constant Field Values hibernate.jta.prefer\_user\_transaction When enabled, specifies that the UserTransaction should be used in preference to the TransactionManager for JTA transaction management. By default, the TransactionManager is preferred. Since: 5.0 See Also: JtaPlatform.retrieveUserTransaction(), JtaPlatform.retrieveTransactionManager(), SessionFactoryBuilder.applyPreferUserTransactions(boolean), Constant Field Values hibernate.jta.track\_by\_thread A transaction can be rolled back by another thread ("tracking by thread") — not the original application. Examples of this include a JTA transaction timeout handled by a background reaper thread. The ability to handle this situation requires checking the Thread ID every time Session is called. This can certainly have performance considerations. Default is true (enabled). See Also: SessionFactoryBuilder.applyJtaTrackingByThread(boolean), Constant Field Values hibernate.loaded\_classes See Also: Constant Field Values hibernate.loader.delay\_entity\_loader\_creations Controls how entity loaders are created. When true, the default, the loaders are only created on first access; this ensures that all access patterns which are not useful to the application are never instantiated, possibly saving a substantial amount of memory for applications having many entities. The only exception is the loader for LockMode.NONE, which will always be eagerly initialized; this is necessary to detect mapping errors. false indicates that all loaders should be created up front; this will consume more memory but ensures all necessary memory is allocated right away. Since: 5.3 See Also: SessionFactoryBuilder.applyDelayedEntityLoaderCreations(boolean), Constant Field Values hibernate.mapping.default\_list\_semantics Specifies the CollectionClassification to use when Hibernate detects a plural attribute typed as List with no explicit list index configuration. Accepts any of: an instance of CollectionClassification the (case insensitive) name of a CollectionClassification (list e.g.) a Class representing either List or Collection By default, when this property is not set, an attribute of type List is taken to have the semantics of a bag unless it is annotated OrderColumn or ListIndexBase. Since: 6.0 See Also: Bag, Constant Field Values hibernate.mapping.precedence Deprecated, for removal: This API element is subject to removal in a future version. hbm.xml mappings are no longer supported, making this attribute irrelevant Specifies the order in which metadata sources should be processed, is a delimited list of values defined by MetadataSourceType. The default is "hbm,class" which that hbm.xml files should be processed first, followed by annotations (combined with orm.xml mappings). See Also: MetadataSourceType, MetadataBuilder.applySourceProcessOrdering(MetadataSourceType…), Constant Field Values hibernate.max\_fetch\_depth Specifies the maximum depth of nested outer join fetching. See Also: SessionFactoryBuilder.applyMaximumFetchDepth(int), Constant Field Values hibernate.metadata\_builder\_contributor Deprecated, for removal: This API element is subject to removal in a future version. Use discovery instead. Names a MetadataBuilderContributor See Also: Constant Field Values hibernate.multi\_tenant\_connection\_provider Specifies a MultiTenantConnectionProvider to use. Since MultiTenantConnectionProvider is also a service, it may be configured directly via the StandardServiceRegistryBuilder. Since: 4.1 See Also: Constant Field Values hibernate.order\_by.default\_null\_ordering Specifies the default precedence of null values in the HQL ORDER BY clause, either none, first, or last. The default is none. See Also: NullPrecedence, SessionFactoryBuilder.applyDefaultNullPrecedence(NullPrecedence), Constant Field Values hibernate.order\_inserts Enable ordering of insert statements by primary key value, for the purpose of more efficient JDBC batching. See Also: SessionFactoryBuilder.applyOrderingOfInserts(boolean), Constant Field Values hibernate.order\_updates Enable ordering of update statements by primary key value, for the purpose of more efficient JDBC batching See Also: SessionFactoryBuilder.applyOrderingOfUpdates(boolean), Constant Field Values hibernate.orm\_xml\_files See Also: Constant Field Values hibernate.persistenceUnitName Specifies the name of the persistence unit. See Also: Constant Field Values hibernate.physical\_naming\_strategy Specifies the PhysicalNamingStrategy to use. By default, PhysicalNamingStrategyStandardImpl is used, in which case physical names are taken to be identical to logical names. Since: 5.0 See Also: MetadataBuilder.applyPhysicalNamingStrategy(org.hibernate.boot.model.naming.PhysicalNamingStrategy), Constant Field Values hibernate.proxool A setting prefix used to indicate settings that target the hibernate-proxool integration See Also: Constant Field Values hibernate.proxool.existing\_pool Proxool property to configure the Proxool Provider from an already existing pool (true / false) See Also: Constant Field Values hibernate.proxool.pool\_alias Proxool property with the Proxool pool alias to use (Required for PROXOOL\_EXISTING\_POOL, PROXOOL\_PROPERTIES, or PROXOOL\_XML) See Also: Constant Field Values hibernate.proxool.properties Proxool property to configure the Proxool provider using a properties file (/path/to/proxool.properties) See Also: Constant Field Values hibernate.proxool.xml Proxool property to configure the Proxool provider using an XML (/path/to/file.xml) See Also: Constant Field Values hibernate.query.fail\_on\_pagination\_over\_collection\_fetch When pagination is used in combination with a fetch join applied to a collection or many-valued association, the limit must be applied in-memory instead of on the database. This typically has terrible performance characteristics, and should be avoided. When enabled, this setting specifies that an exception should be thrown for any query which would result in the limit being applied in-memory. By default, the exception is disabled, and the possibility of terrible performance is left as a problem for the client to avoid. Since: 5.2.13 See Also: Constant Field Values hibernate.query.hql.translator Specifies a HqlTranslator to use for HQL query translation. See Also: Constant Field Values hibernate.query.immutable\_entity\_update\_query\_handling\_mode This setting defines how Immutable entities are handled when executing a bulk update query. Valid options are enumerated by ImmutableEntityUpdateQueryHandlingMode: "warning" specifies that a warning log message is issued when an immutable entity is to be updated via a bulk update statement, and "exception" specifies that a HibernateException should be thrown. By default, a warning is logged. Since: 5.2.17 See Also: ImmutableEntityUpdateQueryHandlingMode, Constant Field Values hibernate.query.in\_clause\_parameter\_padding Determines how parameters occurring in a SQL IN predicate are expanded. By default, the IN predicate expands to include sufficient bind parameters to accommodate the specified arguments. However, for database systems supporting execution plan caching, there’s a better chance of hitting the cache if the number of possible IN clause parameter list lengths is smaller. When this setting is enabled, we expand the number of bind parameters to an integer power of two: 4, 8, 16, 32, 64. Thus, if 5, 6, or 7 arguments are bound to a parameter, a SQL statement with 8 bind parameters in the IN clause will be used, and null will be bound to the left-over parameters. Since: 5.2.17 See Also: Constant Field Values hibernate.query.insert\_strategy Defines the "global" strategy to use for handling HQL and Criteria insert queries. Specifies a SqmMultiTableInsertStrategy. See Also: Constant Field Values hibernate.query.jpaql\_strict\_compliance See Also: Constant Field Values hibernate.query.mutation\_strategy Defines the "global" strategy to use for handling HQL and Criteria mutation queries. Specifies a SqmMultiTableMutationStrategy.. See Also: Constant Field Values hibernate.query.plan\_cache\_enabled When enabled, specifies that query plans should be cached. By default, the query plan cache is disabled, unless one of the configuration properties "hibernate.query.plan\_cache\_max\_size" or "hibernate.query.plan\_parameter\_metadata\_max\_size" is set. See Also: Constant Field Values hibernate.query.plan\_cache\_max\_size The maximum number of entries in the query plan cache or query interpretation cache. The default maximum is 2048. See Also: QueryPlanCache, Constant Field Values hibernate.query.plan\_parameter\_metadata\_max\_size Deprecated. this setting is not currently used The maximum number of ParameterMetadata instances maintained by the QueryInterpretationCache. See Also: Constant Field Values hibernate.query.proc.callable\_named\_params\_enabled When enabled, specifies that Hibernate should attempt to map parameter names given in a ProcedureCall or StoredProcedureQuery to named parameters of the JDBC CallableStatement. Since: 6.0 See Also: SessionFactoryOptions.isUseOfJdbcNamedParametersEnabled(), Constant Field Values hibernate.query.sqm.translator Specifies a SqmTranslatorFactory to use for HQL query translation. See Also: Constant Field Values hibernate.query.startup\_check When enabled, specifies that named queries be checked during startup. By default, named queries are checked at startup. Mainly intended for use in test environments. See Also: SessionFactoryBuilder.applyNamedQueryCheckingOnStartup(boolean), Constant Field Values hibernate.resource.beans.container Identifies a BeanContainer to be used. Note that for CDI-based containers setting this is not necessary - simply pass the BeanManager to use via CDI\_BEAN\_MANAGER and optionally specify DELAY\_CDI\_ACCESS. This setting useful to integrate non-CDI bean containers such as Spring. Since: 5.3 See Also: Constant Field Values hibernate.schema\_management\_tool Specifies the SchemaManagementTool to use for performing schema management. By default, HibernateSchemaManagementTool is used. Since: 5.0 See Also: Constant Field Values hibernate.schema\_update.unique\_constraint\_strategy Unique columns and unique keys both use unique constraints in most dialects. The schema exporter must create these constraints, but database support for finding existing constraints is extremely inconsistent. Worse, unique constraints without explicit names are assigned names with randomly generated characters. Therefore, select from these strategies: DROP\_RECREATE\_QUIETLY: Attempt to drop, then (re-)create each unique constraint, ignoring any exceptions thrown. This is the default. RECREATE\_QUIETLY: attempt to (re-)create unique constraints, ignoring exceptions thrown if the constraint already existed. SKIP: do not attempt to create unique constraints on a schema update. See Also: Constant Field Values hibernate.session.events.auto Defines a default SessionEventListener to be applied to newly-opened Sessions. See Also: Constant Field Values hibernate.session.events.log Controls whether session metrics should be logged for any session in which statistics are being collected. By default, logging of session metrics is disabled unless GENERATE\_STATISTICS is enabled. See Also: Constant Field Values hibernate.session.events.log.LOG\_QUERIES\_SLOWER\_THAN\_MS Specifies a duration in milliseconds defining the minimum query execution time that characterizes a "slow" query. Any SQL query which takes longer than this amount of time to execute will be logged. A value of 0, the default, disables logging of "slow" queries. See Also: Constant Field Values hibernate.session\_factory.interceptor Specifies an Interceptor implementation associated with the SessionFactory and propagated to each Session created from the SessionFactory. Either: an instance of Interceptor, a Class representing a class that implements Interceptor, or the name of a class that implements Interceptor. This setting identifies an Interceptor which is effectively a singleton across all the sessions opened from the SessionFactory to which it is applied; the same instance will be passed to each Session. If there should be a separate instance of Interceptor for each Session, use SESSION\_SCOPED\_INTERCEPTOR instead. Since: 5.0 See Also: SessionFactoryBuilder.applyInterceptor(Interceptor), Constant Field Values hibernate.session\_factory.session\_scoped\_interceptor Specifies an Interceptor implementation associated with the SessionFactory and propagated to each Session created from the SessionFactory. Either: a Class representing a class that implements Interceptor, the name of a class that implements Interceptor, or an instance of Supplier used to obtain the interceptor. Note that this setting cannot specify an Interceptor instance. This setting identifies an Interceptor implementation that is to be applied to every Session opened from the SessionFactory, but unlike INTERCEPTOR, a separate instance created for each Session. Since: 5.2 See Also: SessionFactoryBuilder.applyStatelessInterceptor(Class), SessionFactoryBuilder.applyStatelessInterceptor(Supplier), Constant Field Values hibernate.session\_factory.statement\_inspector Specifies a StatementInspector implementation associated with the SessionFactory, either: an instance of StatementInspector, a Class representing an class that implements StatementInspector, or the name of a class that implements StatementInspector. Since: 5.0 See Also: SessionFactoryBuilder.applyStatementInspector(StatementInspector), Constant Field Values hibernate.session\_factory\_name Setting used to name the Hibernate SessionFactory. Naming the SessionFactory allows for it to be properly serialized across JVMs as long as the same name is used on each JVM. If SESSION\_FACTORY\_NAME\_IS\_JNDI is set to true, this is also the name under which the SessionFactory is bound into JNDI on startup and from which it can be obtained from JNDI. See Also: SESSION\_FACTORY\_NAME\_IS\_JNDI, SessionFactoryRegistry, SessionFactoryBuilder.applyName(String), Constant Field Values hibernate.session\_factory\_name\_is\_jndi Does the value defined by SESSION\_FACTORY\_NAME represent a JNDI namespace into which the SessionFactory should be bound and made accessible? Defaults to true for backwards compatibility. Set this to false if naming a SessionFactory is needed for serialization purposes, but no writable JNDI context exists in the runtime environment or if the user simply does not want JNDI to be used. See Also: SESSION\_FACTORY\_NAME, SessionFactoryBuilder.applyNameAsJndiName(boolean), Constant Field Values hibernate.session\_factory\_observer Specifies a class which implements SessionFactoryObserver and has a constructor with no parameters. See Also: SessionFactoryBuilder.addSessionFactoryObservers(SessionFactoryObserver…), Constant Field Values hibernate.show\_sql Enables logging of generated SQL to the console. See Also: Constant Field Values hibernate.statistics.query\_max\_size This setting controls the number of QueryStatistics entries that will be stored by the Hibernate Statistics object. The default value is 5000. Since: 5.4 See Also: Constant Field Values hibernate.strategy\_registration\_provider Names a StrategyRegistrationProviderList See Also: Constant Field Values hibernate.synonyms If enabled, allows schema update and validation to support synonyms. Due to the possibility that this would return duplicate tables (especially in Oracle), this is disabled by default. See Also: Constant Field Values hibernate.tenant\_identifier\_resolver Specifies a CurrentTenantIdentifierResolver to use, either: an instance of CurrentTenantIdentifierResolver, a Class representing an class that implements CurrentTenantIdentifierResolver, or the name of a class that implements CurrentTenantIdentifierResolver. Since: 4.1 See Also: SessionFactoryBuilder.applyCurrentTenantIdentifierResolver(CurrentTenantIdentifierResolver), Constant Field Values hibernate.timezone.default\_storage Specifies the default strategy for storage of the timezone information for the zoned datetime types OffsetDateTime and ZonedDateTime. The possible options for this setting are enumerated by TimeZoneStorageType. The default is DEFAULT, which guarantees that the instant represented by a zoned datetime type is preserved by a round trip to the database. It does not guarantee that the time zone or offset is preserved. For backward compatibility with older versions of Hibernate, set this property to NORMALIZE. The default strategy specified using this setting may be overridden using the annotation TimeZoneStorage. Since: 6.0 See Also: TimeZoneStorageType, TimeZoneStorage, Constant Field Values hibernate.transaction.auto\_close\_session When enabled, specifies that the Session should be closed automatically at the end of each transaction. See Also: SessionFactoryBuilder.applyAutoClosing(boolean), Constant Field Values hibernate.transaction.coordinator\_class Specify the TransactionCoordinatorBuilder implementation to use for creating instances of TransactionCoordinator, either: an instance of TransactionCoordinatorBuilder, a Class representing a class that implements TransactionCoordinatorBuilder, or the name of a class that implements TransactionCoordinatorBuilder. Since: 5.0 See Also: Constant Field Values hibernate.transaction.flush\_before\_completion When enabled, specifies that automatic flushing should occur during the JTA Synchronization.beforeCompletion() callback. See Also: SessionFactoryBuilder.applyAutoFlushing(boolean), Constant Field Values hibernate.transaction.jta.platform Specifies the JtaPlatform implementation to use for integrating with JTA, either: an instance of JtaPlatform, or the name of a class that implements JtaPlatform. Since: 4.0 See Also: JTA\_PLATFORM\_RESOLVER, Constant Field Values hibernate.transaction.jta.platform\_resolver Specifies a JtaPlatformResolver implementation that should be used to obtain an instance of JtaPlatform. Since: 4.3 See Also: JTA\_PLATFORM, Constant Field Values hibernate.transform\_hbm\_xml.enabled Enables processing hbm.xml mappings by transforming them to mapping.xml and using that processor. Default is false, must be opted-into. Since: 6.1 See Also: Constant Field Values hibernate.transform\_hbm\_xml.unsupported\_feature\_handling How features in a hbm.xml file which are not supported for transformation should be handled. Default is UnsupportedFeatureHandling.ERROR Since: 6.1 See Also: UnsupportedFeatureHandling, Constant Field Values hibernate.type.json\_format\_mapper Specifies a FormatMapper used for JSON serialization and deserialization, either: an instance of FormatMapper, a Class representing a class that implements FormatMapper, the name of a class that implements FormatMapper, or one of the shorthand constants jackson or jsonb. By default, the first of the possible providers that is available at runtime is used, according to the listing order. Since: 6.0 See Also: Constant Field Values hibernate.type.preferred\_boolean\_jdbc\_type Specifies the preferred JDBC type for storing boolean values. When no type is explicitly specified, a sensible dialect-specific default type code is used. Can be overridden locally using JdbcType, JdbcTypeCode, and friends. Can also specify the name of the SqlTypes constant field, for example, hibernate.type.preferred\_boolean\_jdbc\_type=BIT. Since: 6.0 See Also: Constant Field Values hibernate.type.preferred\_duration\_jdbc\_type The preferred JDBC type to use for storing duration values. Falls back to SqlTypes.INTERVAL\_SECOND. Can be overridden locally using JdbcType, JdbcTypeCode, and friends. Can also specify the name of the SqlTypes constant field, for example, hibernate.type.preferred\_duration\_jdbc\_type=NUMERIC. Since: 6.0 See Also: Constant Field Values hibernate.type.preferred\_instant\_jdbc\_type Specifies the preferred JDBC type for storing instant values. When no type is explicitly specified, SqlTypes.TIMESTAMP\_UTC is used. Can be overridden locally using JdbcType, JdbcTypeCode, and friends. Can also specify the name of the SqlTypes constant field, for example, hibernate.type.preferred\_instant\_jdbc\_type=TIMESTAMP. Since: 6.0 See Also: Constant Field Values hibernate.type.preferred\_uuid\_jdbc\_type The preferred JDBC type to use for storing UUID values. Can be overridden locally using JdbcType, JdbcTypeCode, and friends. Can also specify the name of the SqlTypes constant field, for example, hibernate.type.preferred\_uuid\_jdbc\_type=CHAR. Since: 6.0 See Also: Constant Field Values hibernate.type.wrapper\_array\_handling Configurable control over how to handle Byte[] and Character[] types encountered in the application domain model. Allowable semantics are defined by WrapperArrayHandling. Accepted values include: WrapperArrayHandling instance case-insensitive name of a WrapperArrayHandling instance (e.g. allow) Since: 6.2 See Also: Constant Field Values hibernate.type.xml\_format\_mapper Specifies a FormatMapper used for XML serialization and deserialization, either: an instance of FormatMapper, a Class representing a class that implements FormatMapper, the name of a class that implements FormatMapper, or one of the shorthand constants jackson or jaxb. By default, the first of the possible providers that is available at runtime is used, according to the listing order. Since: 6.0.1 See Also: Constant Field Values hibernate.type\_contributors Deprecated, for removal: This API element is subject to removal in a future version. Consider using discovery instead to dynamically locate contributors. Names a TypeContributorList See Also: Constant Field Values hibernate.use\_entity\_where\_clause\_for\_collections Deprecated, for removal: This API element is subject to removal in a future version. Originally added as a backwards compatibility flag The Where annotation specifies a restriction on the table rows which are visible as entity class instances or collection elements. This setting controls whether the restriction applied to an entity should be applied to association fetches (one-to-one, many-to-one, one-to-many and many-to-many) targeting the entity. See Also: Constant Field Values "API Note:" The setting is very misnamed - it applies across all entity associations, not just collections. "Implementation Specification:" Enabled (true) by default, meaning the restriction is applied. When this setting is disabled (false), the restriction is not applied. hibernate.use\_identifier\_rollback When enabled, specifies that the generated identifier of an entity is unset when the entity is deleted. By default, generated identifiers are never unset. See Also: SessionFactoryBuilder.applyIdentifierRollbackSupport(boolean), Constant Field Values hibernate.use\_nationalized\_character\_data By default, Hibernate maps character data represented by Strings and Clobs to the JDBC types Types.VARCHAR and Types.CLOB. This setting, when enabled, turns on the use of explicit nationalized character support for mappings involving character data, specifying that the JDBC types Types.NVARCHAR and Types.NCLOB should be used instead. This setting is relevant for use with databases with explicit nationalization support, and it is not needed for databases whose native varchar and clob types support Unicode data. (If you’re not sure how your database handles Unicode, check out the implementation of Dialect.getNationalizationSupport() for its SQL dialect.) Enabling this setting has two effects: when interacting with JDBC, Hibernate uses operations like PreparedStatement.setNString(int, String) PreparedStatement.setNClob(int, java.sql.NClob) to pass character data, and when generating DDL, the schema export tool uses nchar, nvarchar, or nclob as the generated column type when no column type is explicitly specified using Column.columnDefinition(). This setting is disabled by default, and so Unicode character data may not be persisted correctly for databases with explicit nationalization support. This is a global setting applying to all mappings associated with a given SessionFactory. The Nationalized annotation may be used to selectively enable nationalized character support for specific columns. See Also: MetadataBuilder.enableGlobalNationalizedCharacterDataSupport(boolean), NationalizationSupport, Nationalized, Constant Field Values hibernate.use\_sql\_comments Specifies that comments should be added to the generated SQL. See Also: SessionFactoryBuilder.applySqlComments(boolean), Constant Field Values hibernate.validate\_xml Whether XML should be validated against their schema as Hibernate reads them. Default is true Since: 6.1 See Also: Constant Field Values hibernate.xml\_mapping\_enabled When disabled, specifies that processing of XML-based mappings should be skipped. This is a performance optimization appropriate when all O/R mappings are defined exclusively using annotations. By default, the XML-based mappings are taken into account. Since: 5.4.1 See Also: Constant Field Values jakarta.persistence.bean.manager Used to pass a CDI BeanManager to Hibernate. According to the JPA specification, the BeanManager should be passed at boot time and be ready for immediate use at that time. But not all environments can do this (WildFly, for example). To accommodate such environments, Hibernate provides two options: A proprietary CDI extension SPI (which has been proposed to the CDI spec group as a standard option) which can be used to provide delayed BeanManager access: to use this solution, the reference passed as the BeanManager during bootstrap should be typed as ExtendedBeanManager. Delayed access to the BeanManager reference: here, Hibernate will not access the reference passed as the BeanManager during bootstrap until it is first needed. Note, however, that this has the effect of delaying the detection of any deployment problems until after bootstrapping. This setting is used to configure access to the BeanManager, either directly, or via ExtendedBeanManager. See Also: SessionFactoryBuilder.applyBeanManager(Object), Constant Field Values jakarta.persistence.cache.retrieveMode Set a default value for SpecHints.HINT\_SPEC\_CACHE\_RETRIEVE\_MODE, used when the hint is not explicitly specified. It does not usually make sense to change the default from CacheRetrieveMode.USE. See Also: SpecHints.HINT\_SPEC\_CACHE\_RETRIEVE\_MODE, Constant Field Values jakarta.persistence.cache.storeMode Set a default value for SpecHints.HINT\_SPEC\_CACHE\_STORE\_MODE, used when the hint is not explicitly specified. It does not usually make sense to change the default from CacheStoreMode.USE. See Also: SpecHints.HINT\_SPEC\_CACHE\_RETRIEVE\_MODE, Constant Field Values jakarta.persistence.create-database-schemas The JPA variant of HBM2DDL\_CREATE\_NAMESPACES used to specify whether database schemas used in the mapping model should be created on export in addition to creating the tables, sequences, etc. The default is false, meaning to not create schemas See Also: Constant Field Values jakarta.persistence.database-major-version Used in conjunction with "jakarta.persistence.database-product-name" for the purpose of determining the Dialect to use when the name does not provide enough detail. The value is expected to match what would be returned from DatabaseMetaData.getDatabaseMajorVersion()) for the underlying database. See Also: JAKARTA\_HBM2DDL\_DB\_NAME, Constant Field Values jakarta.persistence.database-minor-version Used in conjunction with "jakarta.persistence.database-product-name" for the purpose of determining the Dialect to use when the name does not provide enough detail. The value is expected to match what would be returned from DatabaseMetaData.getDatabaseMinorVersion()) for the underlying database. See Also: JAKARTA\_HBM2DDL\_DB\_NAME, Constant Field Values jakarta.persistence.database-product-name Specifies the name of the database vendor (as would be reported by DatabaseMetaData.getDatabaseProductName()) for the purpose of determining the Dialect to use. For cases when the name of the database vendor is not enough alone, a combination of "jakarta.persistence.database-product-version", "jakarta.persistence.database-major-version" "jakarta.persistence.database-minor-version" can be used instead See Also: JAKARTA\_HBM2DDL\_DB\_VERSION, JAKARTA\_HBM2DDL\_DB\_MAJOR\_VERSION, JAKARTA\_HBM2DDL\_DB\_MINOR\_VERSION, Constant Field Values "Implementation Specification:" database actions are not available when supplying just the name and versions jakarta.persistence.database-product-version Used in conjunction with "jakarta.persistence.database-product-name" for the purpose of determining the Dialect to use when the name does not provide enough detail. The value is expected to match what would be returned from DatabaseMetaData.getDatabaseProductVersion()) for the underlying database. See Also: JAKARTA\_HBM2DDL\_DB\_NAME, Constant Field Values jakarta.persistence.jdbc.driver Specifies the name of a JDBC driver to use to connect to the database. Used in conjunction with JPA\_JDBC\_URL, JPA\_JDBC\_USER and JPA\_JDBC\_PASSWORD to specify how to connect to the database. When connections are obtained from a DataSource, use either JPA\_JTA\_DATASOURCE or JPA\_NON\_JTA\_DATASOURCE instead. See section 8.2.1.9 See Also: Constant Field Values jakarta.persistence.jdbc.password Specifies the password to use when connecting via JDBC. Used in conjunction with JPA\_JDBC\_DRIVER, JPA\_JDBC\_URL and JPA\_JDBC\_USER to specify how to connect to the database. See JPA 2 section 8.2.1.9 See Also: Constant Field Values jakarta.persistence.jdbc.url Specifies the JDBC connection URL to use to connect to the database. Used in conjunction with JPA\_JDBC\_DRIVER, JPA\_JDBC\_USER and JPA\_JDBC\_PASSWORD to specify how to connect to the database. When connections are obtained from a DataSource, use either JPA\_JTA\_DATASOURCE or JPA\_NON\_JTA\_DATASOURCE instead. See section 8.2.1.9 See Also: Constant Field Values jakarta.persistence.jdbc.user Specifies the database user to use when connecting via JDBC. Used in conjunction with JPA\_JDBC\_DRIVER, JPA\_JDBC\_URL and JPA\_JDBC\_PASSWORD to specify how to connect to the database. See section 8.2.1.9 See Also: Constant Field Values jakarta.persistence.jtaDataSource Specifies the JNDI name of a JTA DataSource. See JPA 2 sections 9.4.3 and 8.2.1.5 See Also: Constant Field Values jakarta.persistence.lock.scope Set a default value for the hint SpecHints.HINT\_SPEC\_LOCK\_SCOPE, used when the hint is not explicitly specified. See JPA 2 sections 8.2.1.9 and 3.4.4.3 See Also: SpecHints.HINT\_SPEC\_LOCK\_SCOPE, Constant Field Values jakarta.persistence.lock.timeout Set a default value for the hint SpecHints.HINT\_SPEC\_LOCK\_TIMEOUT, used when the hint is not explicitly specified. See JPA 2 sections 8.2.1.9 and 3.4.4.3 See Also: SpecHints.HINT\_SPEC\_LOCK\_TIMEOUT, Constant Field Values jakarta.persistence.nonJtaDataSource Specifies the JNDI name of a non-JTA DataSource. See JPA 2 sections 9.4.3 and 8.2.1.5 See Also: Constant Field Values jakarta.persistence.provider Specifies a class implementing PersistenceProvider. Naturally, this should always be HibernatePersistenceProvider, which is the best damn persistence provider ever. There’s no need to explicitly specify this setting when there are no inferior persistence providers floating about. See JPA 2 sections 9.4.3 and 8.2.1.4 See Also: Constant Field Values jakarta.persistence.schema-generation-connection Allows passing a specific Connection instance to be used by SchemaManagementTool for the purpose of determining the Dialect, and for performing database actions if requested. For Dialect resolution, "jakarta.persistence.database-product-name" and, optionally, "jakarta.persistence.database-product-version", "jakarta.persistence.database-major-version", and "jakarta.persistence.database-minor-version" can be used instead See Also: JAKARTA\_HBM2DDL\_DB\_NAME, JAKARTA\_HBM2DDL\_DB\_VERSION, JAKARTA\_HBM2DDL\_DB\_MAJOR\_VERSION, JAKARTA\_HBM2DDL\_DB\_MINOR\_VERSION, Constant Field Values jakarta.persistence.schema-generation.create-script-source Specifies the CREATE script file as either a Reader configured for reading the DDL script file or a string designating a file URL for the DDL script. Hibernate historically also accepted HBM2DDL\_IMPORT\_FILES for a similar purpose. This setting is now preferred. See Also: JAKARTA\_HBM2DDL\_CREATE\_SOURCE, HBM2DDL\_IMPORT\_FILES, Constant Field Values jakarta.persistence.schema-generation.create-source Specifies whether schema generation commands for schema creation are to be determined based on object/relational mapping metadata, DDL scripts, or a combination of the two. See SourceType for the list of legal values. If no value is specified, a default is inferred as follows: if source scripts are specified via "jakarta.persistence.schema-generation.create-source", then "script" is assumed, or otherwise, "metadata" is assumed. See Also: SourceType, Constant Field Values jakarta.persistence.schema-generation.database.action Specifies what type of schema tooling action should be performed against the database specified using either "jakarta.persistence.schema-generation-connection" or the configured ConnectionProvider for the SessionFactory. Valid options are enumerated by Action. This setting takes precedence over "hibernate.hbm2ddl.auto". If no value is specified, the default is "none". See Also: Action, JAKARTA\_HBM2DDL\_CONNECTION, JAKARTA\_JDBC\_URL, Constant Field Values jakarta.persistence.schema-generation.drop-script-source Specifies the DROP script file as either a Reader configured for reading the DDL script file or a string designating a file URL for the DDL script. See Also: JAKARTA\_HBM2DDL\_DROP\_SOURCE, Constant Field Values jakarta.persistence.schema-generation.drop-source Specifies whether schema generation commands for schema dropping are to be determined based on object/relational mapping metadata, DDL scripts, or a combination of the two. See SourceType for the list of legal values. If no value is specified, a default is inferred as follows: if source scripts are specified via "jakarta.persistence.schema-generation.drop-script-source", then "script" is assumed, or otherwise, "metadata" is assumed. See Also: SourceType, Constant Field Values jakarta.persistence.schema-generation.scripts.action Specifies what type of schema tooling action should be written to script files. Valid options are enumerated by Action. The script file is identified using "jakarta.persistence.schema-generation.scripts.create-target". If no value is specified, the default is "none". See Also: Action, JAKARTA\_HBM2DDL\_SCRIPTS\_CREATE\_TARGET, JAKARTA\_HBM2DDL\_SCRIPTS\_DROP\_TARGET, Constant Field Values jakarta.persistence.schema-generation.scripts.create-target For cases where "jakarta.persistence.schema-generation.scripts.action" indicates that schema creation commands should be written to a script file, this setting specifies either a Writer configured for output of the DDL script or a string specifying the file URL for the DDL script. See Also: JAKARTA\_HBM2DDL\_SCRIPTS\_ACTION, Constant Field Values jakarta.persistence.schema-generation.scripts.drop-target For cases where "jakarta.persistence.schema-generation.scripts.action" indicates that schema drop commands should be written to a script file, this setting specifies either a Writer configured for output of the DDL script or a string specifying the file URL for the DDL script. See Also: JAKARTA\_HBM2DDL\_SCRIPTS\_ACTION, Constant Field Values jakarta.persistence.sharedCache.mode When enabled, specifies that the second-level cache (which JPA calls the "shared" cache) may be used, as per the rules defined in JPA 2 section 3.1.7. See JPA 2 sections 9.4.3 and 8.2.1.7 See Also: SharedCacheMode, Constant Field Values jakarta.persistence.sql-load-script-source JPA-standard variant of HBM2DDL\_IMPORT\_FILES for specifying a database initialization script to be run as part of schema-export Specifies a Reader configured for reading of the SQL load script or a string designating the URL for the SQL load script. See Also: Constant Field Values jakarta.persistence.transactionType Specifies the type of transactions supported by the entity managers. The default depends on whether the program is considered to be executing in a Java SE or EE environment: For Java SE, the default is RESOURCE\_LOCAL. For Java EE, the default is JTA. See JPA 2 sections 9.4.3 and 8.2.1.2 See Also: Constant Field Values jakarta.persistence.validation.factory Used to pass along any discovered ValidatorFactory. See Also: SessionFactoryBuilder.applyValidatorFactory(Object), Constant Field Values jakarta.persistence.validation.group.pre-persist Used to coordinate with bean validators. See JPA 2 section 8.2.1.9 See Also: Constant Field Values jakarta.persistence.validation.group.pre-remove Used to coordinate with bean validators. See JPA 2 section 8.2.1.9 See Also: Constant Field Values jakarta.persistence.validation.group.pre-update Used to coordinate with bean validators. See JPA 2 section 8.2.1.9 See Also: Constant Field Values jakarta.persistence.validation.mode Indicates which form of automatic validation is in effect as per the rules defined in JPA 2 section 3.6.1.1. See JPA 2 sections 9.4.3 and 8.2.1.8 See Also: ValidationMode, Constant Field Values javax.persistence.bean.manager Deprecated. Use JAKARTA\_CDI\_BEAN\_MANAGER instead Used to pass a CDI BeanManager to Hibernate. According to the JPA specification, the BeanManager should be passed at boot time and be ready for immediate use at that time. But not all environments can do this (WildFly, for example). To accommodate such environments, Hibernate provides two options: A proprietary CDI extension SPI (which has been proposed to the CDI spec group as a standard option) which can be used to provide delayed BeanManager access: to use this solution, the reference passed as the BeanManager during bootstrap should be typed as ExtendedBeanManager. Delayed access to the BeanManager reference: here, Hibernate will not access the reference passed as the BeanManager during bootstrap until it is first needed. Note, however, that this has the effect of delaying the detection of any deployment problems until after bootstrapping. This setting is used to configure access to the BeanManager, either directly, or via ExtendedBeanManager. See Also: Constant Field Values javax.persistence.cache.retrieveMode Deprecated. Use JAKARTA\_SHARED\_CACHE\_RETRIEVE\_MODE instead Used to indicate if the provider should attempt to retrieve requested data in the shared cache. See Also: CacheRetrieveMode, Constant Field Values "API Note:" This is not a legal property for an EntityManagerFactory. javax.persistence.cache.storeMode Deprecated. Use JAKARTA\_SHARED\_CACHE\_STORE\_MODE instead Used to indicate if the provider should attempt to store data loaded from the database in the shared cache. See Also: CacheStoreMode, Constant Field Values "API Note:" This is not a legal property for an EntityManagerFactory. javax.persistence.create-database-schemas Deprecated. Use JAKARTA\_HBM2DDL\_CREATE\_SCHEMAS instead See Also: Constant Field Values javax.persistence.database-major-version Deprecated. Use JAKARTA\_HBM2DDL\_DB\_MAJOR\_VERSION instead Specifies the major version of the underlying database, as would be returned by DatabaseMetaData.getDatabaseMajorVersion() for the target database. This value is used to help more precisely determine how to perform schema generation tasks for the database in cases where "javax.persistence.database-product-name" does not provide enough distinction. See Also: DIALECT\_DB\_NAME, DIALECT\_DB\_MINOR\_VERSION, Constant Field Values javax.persistence.database-minor-version Deprecated. Use JAKARTA\_HBM2DDL\_DB\_MINOR\_VERSION instead Specifies the minor version of the underlying database, as would be returned by DatabaseMetaData.getDatabaseMinorVersion() for the target database. This setting is used in Dialect resolution. See Also: DIALECT\_DB\_NAME, DIALECT\_DB\_MAJOR\_VERSION, DialectResolver, Constant Field Values javax.persistence.database-product-name Deprecated. Use JAKARTA\_HBM2DDL\_DB\_NAME instead Specifies the name of the database provider in cases where a connection to the database is not available (usually for generating scripts). In such cases, a value for this setting must be specified. The value of this setting is expected to match the value returned by DatabaseMetaData.getDatabaseProductName() for the target database. Additionally, specifying "javax.persistence.database-major-version", and perhaps even "javax.persistence.database-minor-version", may be required for high quality DDL generation. See Also: DIALECT\_DB\_VERSION, DIALECT\_DB\_MAJOR\_VERSION, DIALECT\_DB\_MINOR\_VERSION, Constant Field Values javax.persistence.database-product-version Deprecated. Use JAKARTA\_HBM2DDL\_DB\_VERSION instead Specifies the name of the database provider in cases where a connection to the database is not available (usually for generating scripts). This value is used to help more precisely determine how to perform schema generation tasks for the underlying database in cases where "javax.persistence.database-product-name" does not provide enough distinction. The value of this setting is expected to match the value returned by DatabaseMetaData.getDatabaseProductVersion() for the target database. See Also: DIALECT\_DB\_NAME, Constant Field Values javax.persistence.jdbc.driver Deprecated. Use JAKARTA\_JDBC\_DRIVER instead The name of a JDBC driver to use to connect to the database. Used in conjunction with JPA\_JDBC\_URL, JPA\_JDBC\_USER and JPA\_JDBC\_PASSWORD to specify how to connect to the database. When connections are obtained from a DataSource, use either JPA\_JTA\_DATASOURCE or JPA\_NON\_JTA\_DATASOURCE instead. See section 8.2.1.9 See Also: Constant Field Values javax.persistence.jdbc.password Deprecated. Use JAKARTA\_JDBC\_PASSWORD instead The password to use when connecting via JDBC. Used in conjunction with JPA\_JDBC\_DRIVER, JPA\_JDBC\_URL and JPA\_JDBC\_USER to specify how to connect to the database. See JPA 2 section 8.2.1.9 See Also: Constant Field Values javax.persistence.jdbc.url Deprecated. Use JAKARTA\_JDBC\_URL instead The JDBC connection URL to use to connect to the database. Used in conjunction with JPA\_JDBC\_DRIVER, JPA\_JDBC\_USER and JPA\_JDBC\_PASSWORD to specify how to connect to the database. When connections are obtained from a DataSource, use either JPA\_JTA\_DATASOURCE or JPA\_NON\_JTA\_DATASOURCE instead. See section 8.2.1.9 See Also: Constant Field Values javax.persistence.jdbc.user Deprecated. Use JAKARTA\_JDBC\_USER instead The database user to use when connecting via JDBC. Used in conjunction with JPA\_JDBC\_DRIVER, JPA\_JDBC\_URL and JPA\_JDBC\_PASSWORD to specify how to connect to the database. See section 8.2.1.9 See Also: Constant Field Values javax.persistence.jtaDataSource Deprecated. Use JAKARTA\_JTA\_DATASOURCE instead The JNDI name of a JTA DataSource. See JPA 2 sections 9.4.3 and 8.2.1.5 See Also: Constant Field Values javax.persistence.lock.scope Deprecated. Use JAKARTA\_LOCK\_SCOPE instead Used to request (hint) a pessimistic lock scope. See JPA 2 sections 8.2.1.9 and 3.4.4.3 See Also: Constant Field Values javax.persistence.lock.timeout Deprecated. Use JAKARTA\_LOCK\_TIMEOUT instead Used to request (hint) a pessimistic lock timeout (in milliseconds). See JPA 2 sections 8.2.1.9 and 3.4.4.3 See Also: Constant Field Values javax.persistence.nonJtaDataSource Deprecated. Use JAKARTA\_NON\_JTA\_DATASOURCE instead The JNDI name of a non-JTA DataSource. See JPA 2 sections 9.4.3 and 8.2.1.5 See Also: Constant Field Values javax.persistence.provider Deprecated. Use JAKARTA\_PERSISTENCE\_PROVIDER instead Specifies a class implementing PersistenceProvider. See JPA 2 sections 9.4.3 and 8.2.1.4 See Also: Constant Field Values javax.persistence.schema-generation-connection Deprecated. Use JAKARTA\_HBM2DDL\_CONNECTION instead See Also: Constant Field Values javax.persistence.schema-generation.create-script-source Deprecated. Migrate to JAKARTA\_HBM2DDL\_CREATE\_SCRIPT\_SOURCE See Also: Constant Field Values javax.persistence.schema-generation.create-source Deprecated. Migrate to JAKARTA\_HBM2DDL\_CREATE\_SOURCE instead See Also: SourceType, Constant Field Values javax.persistence.schema-generation.database.action Deprecated. Use JAKARTA\_HBM2DDL\_DATABASE\_ACTION instead See Also: Constant Field Values javax.persistence.schema-generation.drop-script-source Deprecated. Migrate to JAKARTA\_HBM2DDL\_DROP\_SCRIPT\_SOURCE See Also: Constant Field Values javax.persistence.schema-generation.drop-source Deprecated. Migrate to JAKARTA\_HBM2DDL\_DROP\_SOURCE. See Also: SourceType, Constant Field Values javax.persistence.schema-generation.scripts.action Deprecated. Use JAKARTA\_HBM2DDL\_SCRIPTS\_ACTION instead See Also: Constant Field Values javax.persistence.schema-generation.scripts.create-target Deprecated. Migrate to JAKARTA\_HBM2DDL\_SCRIPTS\_CREATE\_TARGET See Also: Constant Field Values javax.persistence.schema-generation.scripts.drop-target Deprecated. Migrate to JAKARTA\_HBM2DDL\_SCRIPTS\_DROP\_TARGET See Also: Constant Field Values javax.persistence.sharedCache.mode Deprecated. Use JAKARTA\_SHARED\_CACHE\_MODE instead Used to indicate whether second-level (what JPA terms shared cache) caching is enabled as per the rules defined in JPA 2 section 3.1.7. See JPA 2 sections 9.4.3 and 8.2.1.7 See Also: SharedCacheMode, Constant Field Values javax.persistence.sql-load-script-source Deprecated. Use JAKARTA\_HBM2DDL\_LOAD\_SCRIPT\_SOURCE instead See Also: Constant Field Values javax.persistence.transactionType Deprecated. Use JAKARTA\_TRANSACTION\_TYPE instead The type of transactions supported by the entity managers. See JPA 2 sections 9.4.3 and 8.2.1.2 See Also: Constant Field Values javax.persistence.validation.factory Deprecated. Use JAKARTA\_VALIDATION\_FACTORY instead Used to pass along any discovered validator factory. See Also: Constant Field Values javax.persistence.validation.group.pre-persist Deprecated. Use JAKARTA\_PERSIST\_VALIDATION\_GROUP instead Used to coordinate with bean validators. See JPA 2 section 8.2.1.9 See Also: Constant Field Values javax.persistence.validation.group.pre-remove Deprecated. Use JAKARTA\_REMOVE\_VALIDATION\_GROUP instead Used to coordinate with bean validators. See JPA 2 section 8.2.1.9 See Also: Constant Field Values javax.persistence.validation.group.pre-update Deprecated. Use JAKARTA\_UPDATE\_VALIDATION\_GROUP instead Used to coordinate with bean validators. See JPA 2 section 8.2.1.9 See Also: Constant Field Values javax.persistence.validation.mode Deprecated. Use JAKARTA\_VALIDATION\_MODE instead Used to indicate what form of automatic validation is in effect as per rules defined in JPA 2 section 3.6.1.1. See JPA 2 sections 9.4.3 and 8.2.1.8 See Also: ValidationMode, Constant Field Values org.hibernate.flushMode Deprecated, for removal: This API element is subject to removal in a future version. There are much better ways to control the flush mode of a session, for example, SessionBuilder.flushMode(org.hibernate.FlushMode) or Session.setHibernateFlushMode(org.hibernate.FlushMode). Used to determine flush mode. See Also: HibernateHints.HINT\_FLUSH\_MODE, Constant Field Values 34.2. Hibernate’s entity version (audit/history) support org.hibernate.envers.allow\_identifier\_reuse Guarantees proper validity audit strategy behavior when application reuses identifiers of deleted entities. Exactly one row with null end date exists for each identifier. See Also: Constant Field Values org.hibernate.envers.audit\_strategy Audit strategy. Defaults to DefaultAuditStrategy. See Also: Constant Field Values org.hibernate.envers.audit\_strategy\_validity\_end\_rev\_field\_name Column name that will hold the end revision number in audit entities. Defaults to REVEND. See Also: Constant Field Values org.hibernate.envers.audit\_strategy\_validity\_revend\_timestamp\_field\_name Column name of the timestamp of the end revision until which the data was valid. Defaults to REVEND\_TSTMP. See Also: Constant Field Values org.hibernate.envers.audit\_strategy\_validity\_revend\_timestamp\_legacy\_placement Whether to use legacy validity audit strategy revision end timestamp behavior where the field is not included as part of the joined entity inheritance subclass audit tables. Defaults to true. Since: 6.0 See Also: Constant Field Values org.hibernate.envers.audit\_strategy\_validity\_revend\_timestamp\_numeric Determines whether the timestamp of the end revision is stored as a numeric data type. Defaults to false. Since: 6.0 See Also: Constant Field Values org.hibernate.envers.audit\_strategy\_validity\_store\_revend\_timestamp Store the timestamp of the end revision, until which the data was valid, in addition to the end revision itself. Defaults to false. See Also: Constant Field Values org.hibernate.envers.audit\_table\_prefix Audit table prefix. Empty by default. See Also: Constant Field Values org.hibernate.envers.audit\_table\_suffix Audit table suffix. Defaults to \_AUD. See Also: Constant Field Values org.hibernate.envers.cascade\_delete\_revision Deletion of a revision entity will cause a foreign key constraint database error when at least one audit record exists for that revision. By enabling this feature, deletion of the revision entity will also force all audit records associated to that revision to be deleted via cascade. Defaults to false. Since: 4.3.0 See Also: Constant Field Values org.hibernate.envers.default\_catalog Default name of the catalog containing audit tables. See Also: Constant Field Values org.hibernate.envers.default\_schema Default name of the schema containing audit tables. See Also: Constant Field Values org.hibernate.envers.do\_not\_audit\_optimistic\_locking\_field Treats optimistic locking properties as unversioned. Defaults to true. See Also: Constant Field Values org.hibernate.envers.embeddable\_set\_ordinal\_field\_name Name of column used for storing ordinal of the change in sets of embeddable elements. Defaults to SETORDINAL. See Also: Constant Field Values org.hibernate.envers.find\_by\_revision\_exact\_match Forces AuditReader#find implementations that accept a revision-number argument to perform an exact match against the supplied revision number rather than potentially returning hits that are less-than or equal-to the supplied revision number. This option is meant to maintain backward compatibility while attempting to correct a bug in behavior without impacting existing users who may use the current behavior. Defaults to false. Since: 5.4.4 See Also: Constant Field Values org.hibernate.envers.global\_relation\_not\_found\_legacy\_flag Globally defines whether legacy relation not-found behavior should be used or not. Defaults to true. By specifying true, any EntityNotFoundException will be thrown unless the containing class or property explicitly specifies that use case to be ignored. Conversely, when specifying the value false, the inverse applies and requires explicitly specifying the use case as error so that the exception is thrown. See Also: Constant Field Values org.hibernate.envers.global\_with\_modified\_flag Globally activates modified properties flag feature. Defaults to false. See Also: Constant Field Values org.hibernate.envers.modified\_column\_naming\_strategy Specifies the ModifiedColumnNamingStrategy to use Defaults to LegacyModifiedColumnNamingStrategy. Since: 5.4.7 See Also: Constant Field Values org.hibernate.envers.modified\_flag\_suffix Suffix of modified flag columns. Defaults to \_MOD. See Also: Constant Field Values org.hibernate.envers.original\_id\_prop\_name Original id property name name. Defaults to originalId. See Also: Constant Field Values org.hibernate.envers.revision\_field\_name Revision field name. Defaults to REV. See Also: Constant Field Values org.hibernate.envers.revision\_listener Fully qualified class name of user defined revision listener. See Also: Constant Field Values org.hibernate.envers.revision\_on\_collection\_change Triggers revision generation when not-owned relation field changes. Defaults to true. See Also: Constant Field Values org.hibernate.envers.revision\_sequence\_nocache Whether to apply a nocache configuration for the revision sequence. This is mostly interesting for testing. See Also: Constant Field Values org.hibernate.envers.revision\_type\_field\_name Revision type field name. Defaults to REVTYPE. See Also: Constant Field Values org.hibernate.envers.store\_data\_at\_delete Indicates whether entity data should be stored during removal. Defaults to false. See Also: Constant Field Values org.hibernate.envers.track\_entities\_changed\_in\_revision Track entity names that have been changed during each revision. Defaults to false. See Also: Constant Field Values org.hibernate.envers.use\_revision\_entity\_with\_native\_id Use revision entity with native identifier generator. Defaults to true for backward compatibility. See Also: Constant Field Values 34.3. Integration for javax.cache into Hibernate as a second-level caching service hibernate.javax.cache. See Also: Constant Field Values hibernate.javax.cache.cache\_manager Allows providing hibernate-jcache with a custom JCache CacheManager. See Also: Constant Field Values hibernate.javax.cache.missing\_cache\_strategy Define the behavior of the region factory when a cache is missing, i.e. when the cache was not created by the cache manager as it started. See MissingCacheStrategy for the various possible values. Default value is MissingCacheStrategy.FAIL. See Also: Constant Field Values hibernate.javax.cache.provider Allows providing hibernate-jcache with a custom JCache CachingProvider. See Also: Constant Field Values hibernate.javax.cache.uri Designates the URI for a specific JCache CacheManager JCacheRegionFactory should ask the CachingProvider for See Also: CachingProvider.getCacheManager(URI, ClassLoader), Constant Field Values jcache See Also: Constant Field Values