Full list of JDBC Provider Configuration Keys

Introduction

In most usecases, only the required configuration keys of DataSources will be used: `url`, `username`, `password`, `validationQuery`.

The following chapter provides a full list of supported key name patterns.

Ful list of org.clazzes.jdbc.provider datasource configuration keys

The JDBC-Provider supplies applications with Instances of org.apache.commons.dbcp.BasicDataSource from Apache's commons-dbcp project in Version 1.x, version 2.x of JDBC-Provider uses commons-dbcp2's org.apache.commons.dbcp2.BasicDataSource, therefore most values go right into it's setters.

Key	Description	JDBC-Provider Version
`accessToUnderlyingConnectionAllowed`	Should not matter. Changes have no effect after pool initialization.	1.x and 2.x
`connectionInitSqls`	List of SQL Statements to be executed when a physical connection is first created. Comma separated. Changes have no effect after pool initialization.	1.x and 2.x
`connectionProperties`	Connection properties.	1.x and 2.x
`defaultAutoCommit`	Default autocommit state. Should not matter for well-designed applications. Changes have no effect after pool initialization.	1.x and 2.x
`defaultCatalog`	Default catalog name. Changes have no effect after pool initialization.	1.x and 2.x
`defaultReadOnly`	Changes have no effect after pool initialization.	1.x and 2.x
`defaultTransactionIsolation`	Changes have no effect after pool initialization.	1.x and 2.x
`driverClassName`	Optional for common database backends. Name of the database driver to use. For MySQL, MSSQL, PostgreSQL and Oracle the driver class name is automatically deduced from the JDBC URL. The class is resolved using the class loader of the jdbc-provider bundle. This allows for putting JDBC driver on the boot classloader of the OSGi container or creating fragment bundles, which supply JDBC drivers to the jdbc-provider bundle.	1.x and 2.x
`initialSize`	Initial size of the pool. Changes have no effect after pool initialization.	1.x and 2.x
`maxActive`	Maximum number of active connections that can be allocated at the same time. Use a negative value for no limit.	1.x
`maxTotal`	Maximum number of active connections that can be allocated at the same time.	2.x, if this value is not set, the value of `maxActive` will be taken for compatibility reasons.
`fastFailValidation`	Whether conenctions with previous SQL states indicating a server dconnect are immediately evicted.	2.x
`maxConnLifetimeMillis`	Maximal connection lifetime in milliseconds	2.x
`maxIdle`	Maximum number of connections that can remain idle in the pool.	1.x and 2.x
`maxOpenPreparedStatement`	Maximum number of PreparedStatements to pool. Use a negative value for no limit. Changes have no effect after pool initialization.	1.x and 2.x
`maxWait`	The maximum number of milliseconds that the pool will wait for a connection to be returned before throwing an exception. Use a negative value for no limit.	1.x
`maxWaitMillis`	The same value as `maxWait`, but for Version 2.x	2.x, if this value is not set, the value of `maxWait` will be taken for compatibility reasons.
`minEvictableIdleTimeMillis`	Minimum amount of time an object may sit idle in the pool.	1.x and 2.x

`minIdle`	Minimum number of idle connections in the pool.	1.x and 2.x
`password`	Required. JDBC Password. Changes have no effect after pool initialization.	1.x and 2.x
`poolPreparedStatements`	Whether to pool statements or not. Changes have no effect after pool initialization.	1.x and 2.x
`removeAbandoned`	Whether abandoned (blocking) queries will be removed form the pool. This is a last-resort option, which has the potential to also evict long-running "ordinary" query.	1.x
`removeAbandonedOnBorrow`	Whether abandoned (blocking) queries will be removed form the pool on connection borrow.	2.x
`removeAbandonedOnMaintenance`	Whether abandoned (blocking) queries will be removed form the pool during pool maintenance.	2.x
`removeAbandonedTimeout`	The timeout in second after which a long-running query is considered as abandoned.	1.x and 2.x
`testOnBorrow`	Wether the validation query shall be executed before each "borrow" from the pool.	1.x and 2.x
`testOnReturn`	Wether the validation query shall be executed when a connection is returned to the pool.	1.x and 2.x
`testOnCreate`	Whether the validation query will be run immediately after the conenction creation.	2.x
`testWhileIdle`	Wether the validation query shall be executed regularily to keep the connection alive.	1.x and 2.x
`timeBetweenEvictionRunsMillis`	The time in milliseconds between two runs of the IDLE evictor Runnable.	1.x and 2.x
`url`	Required. JDBC URL. JDBC URL examples for common databases can be found in our JDBC Snippets. Changes have no effect after pool initialization.	1.x and 2.x
`username`	Required. JDBC User. Changes have no effect after pool initialization.	1.x and 2.x
`validationQuery`	Required. Validation query, executed to ensure the application receives a valid connection. Common validation queries can be found in our JDBC Snippets.	1.x and 2.x

For more supported key patterns take a look at the JavaDoc of the org.clazzes.util.jdbc.provider.JdbcProvider class.

Typical JDBC URLs and validation queries can be found in our JDBC Snippets.