Generator

The optional <generator> child element names a Java class used to generate unique identifiers for instances of the persistent class. If any parameters are required to configure or initialize the generator instance, they are passed using the <param> element.

```
<id name="id" type="long" column="cat_id">
        <generator class="org.hibernate.id.TableHiLoGenerator">
                <param name="table">uid_table</param>
                <param name="column">next_hi_value_column</param>
        </generator>
</id>
```

All generators implement the interface `org.hibernate.id.IdentifierGenerator`. This is a very simple interface. Some applications can choose to provide their own specialized implementations, however, Hibernate provides a range of built-in implementations. The shortcut names for the built-in generators are as follows:

`increment`
: generates identifiers of type `long`, `short` or `int` that are unique only when no other process is inserting data into the same table. *Do not use in a cluster.*

`identity`
: supports identity columns in DB2, MySQL, MS SQL Server, Sybase and HypersonicSQL. The returned identifier is of type `long`, `short` or `int`.

`sequence`
: uses a sequence in DB2, PostgreSQL, Oracle, SAP DB, McKoi or a generator in Interbase. The returned identifier is of type `long`, `short` or `int`

`hilo`
: uses a hi/lo algorithm to efficiently generate identifiers of type `long`, `short` or `int`, given a table and column (by default `hibernate_unique_key` and `next_hi` respectively) as a source of hi values. The hi/lo algorithm generates identifiers that are unique only for a particular database.

`seqhilo`
: uses a hi/lo algorithm to efficiently generate identifiers of type `long`, `short` or `int`, given a named database sequence.

`uuid`
: uses a 128-bit UUID algorithm to generate identifiers of type string that are unique within a network (the IP address is used). The UUID is encoded as a string of 32 hexadecimal digits in length.

`guid`
: uses a database-generated GUID string on MS SQL Server and MySQL.

`native`
: selects `identity`, `sequence` or `hilo` depending upon the capabilities of the underlying database.

`assigned`
: lets the application assign an identifier to the object before `save()` is called. This is the default strategy if no <generator> element is specified.

`select`
: retrieves a primary key, assigned by a database trigger, by selecting the row by some unique key and retrieving the primary key value.

`foreign`
: uses the identifier of another associated object. It is usually used in conjunction with a <one-to-one> primary key association.

`sequence-identity`
: a specialized sequence generation strategy that utilizes a database sequence for the actual value generation, but combines this with JDBC3 getGeneratedKeys to return the generated identifier value as part of the insert statement execution. This strategy is only supported on Oracle 10g drivers targeted for JDK 1.4. Comments on these insert statements are disabled due to a bug in the Oracle drivers.