**1.4.1. Customizing the mapping files**

Hibernate provides several elements and attributes to customize your mapping files. They are listed in

[Table 1.3, “Elements and attributes provided for customizing mapping files”](file:///E:\Java\Java%20EE\libs%20and%20other\libs\Hibernate\hibernate-release-4.3.4.Final\documentation\devguide\en-US\html\ch01.html#tab-customizing-schema), and a logical order of customization is presented in [Procedure 1.1, “Customizing the schema”](file:///E:\Java\Java%20EE\libs%20and%20other\libs\Hibernate\hibernate-release-4.3.4.Final\documentation\devguide\en-US\html\ch01.html#proc-customizing-schema).

**Table 1.3. Elements and attributes provided for customizing mapping files**

| **Name** | **Type of value** | **Description** |
| --- | --- | --- |
| length | number | Column length |
| precision | number | Decimal precision of column |
| scale | number | Decimal scale of column |
| not-null | true or false | Whether a column is allowed to hold null values |
| unique | true or false | Whether values in the column must be unique |
| index | string | The name of a multi-column index |
| unique-key | string | The name of a multi-column unique constraint |
| foreign-key | string | The name of the foreign key constraint generated for an association. This applies to <one-to-one>, <many-to-one>, <key>, and <many-to-many> mapping elements. inverse="true" sides are skipped by SchemaExport. |
| sql-type | string | Overrides the default column type. This applies to the <column> element only. |
| default | string | Default value for the column |
| check | string | An SQL check constraint on either a column or atable |

**Procedure 1.1. Customizing the schema**

1. **Set the length, precision, and scale of mapping elements.**

Many Hibernate mapping elements define optional attributes named **length**, **precision**, and **scale**.

<property **name**="zip" **length**="5"/>

<property **name**="balance" **precision**="12" **scale**="2"/>

1. **Set the not-null, UNIQUE, unique-key attributes.**

The **not-null** and **UNIQUE** attributes generate constraints on table columns.

The unique-key attribute groups columns in a single, unique key constraint. The attribute overrides the name of any generated unique key constraint.

<many-to-one **name**="bar" **column**="barId" **not-null**="true"/>

<element **column**="serialNumber" **type**="long" **not-null**="true" **unique**="true"/>

<many-to-one **name**="org" **column**="orgId" **unique-key**="OrgEmployeeId"/>

<property **name**="employeeId" **unique-key**="OrgEmployee"/>

1. **Set the index and foreign-key attributes.**

The **index** attribute specifies the name of an index for Hibernate to create using the mapped column or columns. You can group multiple columns into the same index by assigning them the same index name.

A foreign-key attribute overrides the name of any generated foreign key constraint.

<many-to-one **name**="bar" **column**="barId" **foreign-key**="FKFooBar"/>

1. **Set child <column> elements.**

Many mapping elements accept one or more child <column> elements. This is particularly useful for mapping types involving multiple columns.

<property **name**="name" **type**="my.customtypes.Name"/>

<column **name**="last" **not-null**="true" **index**="bar\_idx" **length**="30"/>

<column **name**="first" **not-null**="true" **index**="bar\_idx" **length**="20"/>

<column **name**="initial"/>

</property>

1. **Set the default attribute.**

The **default** attribute represents a default value for a column. Assign the same value to the mapped property before saving a new instance of the mapped class.

<property **name**="credits" **type**="integer" **insert**="false">

<column **name**="credits" **default**="10"/>

</property>

<version **name**="version" **type**="integer" **insert**="false">

<column **name**="version" **default**="0"/>

</version>

1. **Set the sql-type attribure.**

Use the **sql-type** attribute to override the default mapping of a Hibernate type to SQL datatype.

<property **name**="balance" **type**="float">

<column **name**="balance" **sql-type**="decimal(13,3)"/>

</property>

1. **Set the check attribute.**

use the **check** attribute to specify a *check* constraint.

<property **name**="foo" **type**="integer">

<column **name**="foo" **check**="foo > 10"/>

</**property**>

<class **name**="Foo" **table**="foos" **check**="bar < 100.0">

...

<**property** **name**="bar" **type**="float"/>

</class>

1. **Add <comment> elements to your schema.**

Use the <comment> element to specify comments for the generated schema.

<class **name**="Customer" **table**="CurCust">

<comment>Current customers only</comment>

...

</class>

**1.4.2. Running the SchemaExport tool**

The SchemaExport tool writes a DDL script to standard output, executes the DDL statements, or both.

**Example 1.7. SchemaExport syntax**

java -cp hibernate\_classpaths org.hibernate.tool.hbm2ddl.SchemaExport ***options*** ***mapping\_files***

**Table 1.4. SchemaExport Options**

| **Option** | **Description** |
| --- | --- |
| --quiet | do not output the script to standard output |
| --drop | only drop the tables |
| --create | only create the tables |
| --text | do not export to the database |
| --output=***my\_schema.ddl*** | output the ddl script to a file |
| --naming=***eg.MyNamingStrategy*** | select a NamingStrategy |
| --config=***hibernate.cfg.xml*** | read Hibernate configuration from an XML file |
| --properties=***hibernate.properties*** | read database properties from a file |
| --format | format the generated SQL nicely in the script |
| --delimiter=***;*** | set an end-of-line delimiter for the script |

**Example 1.8. Embedding SchemaExport into your application**

Configuration cfg = ....;

**new** SchemaExport(cfg).create(false, true);