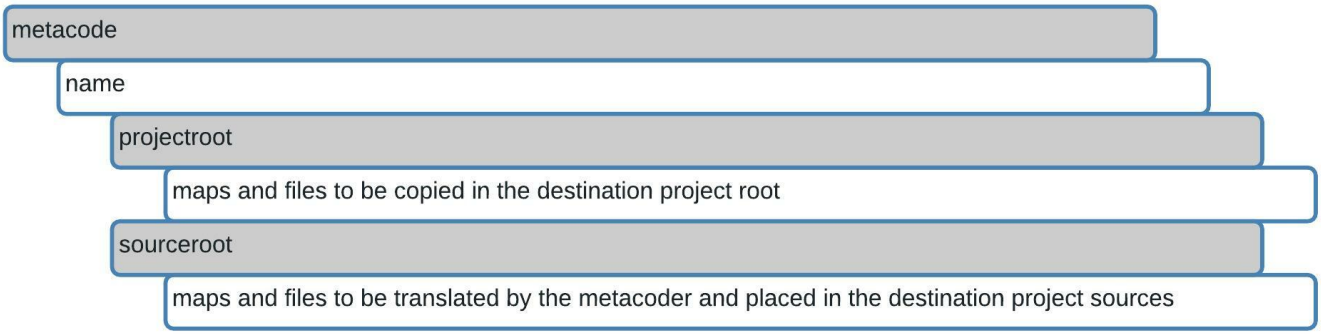
Metacode reference

This manual is a reference to for the template map structure and the template pseudo language

Assumed is knowlegde about relational database structure

Template map structure

maps with grey background have a fixed name



Each metacode project has a map with the project name, stored under the map metacode.

Each metacode map must have 2 submaps

1. projectroot - all content is copied without any modification to the project root map
2. sourceroot - all content is processed by the metacoder and copied to the project source map

The projectroot map typically contains dependencies embedded in the project

The sourceroot map contains the source map structure and code templates

Program language settings

The program languages in the Metacoder are defined by their file extention.

Examples: .java, .js

Each extention you want to be processed by the metacode processor must have a file in the format of “extention.xml”.

These files are stored in the templates map.

The content of these files have a fixed structure.

Example for java files

```
<language>
    <name>java</name>
    <tablenamefilter>
    </tablenamefilter>
    <dbfunctions>
        <java.lang.String>getString(":column:")</java.lang.String>
        <long>getLong(":column:")</long>
        <boolean>getBoolean(":column:")</boolean>
        <java.sql.Date>getDate(":column:")</java.sql.Date>
        <float>getFloat(":column:")</float>
        <double>getDouble(":column:")</double>
        <int>getInt(":column:")</int>
        <java.sql.Time>getTime(":column:")</java.sql.Time>
        <java.sql.Timestamp>getTimestamp(":column:")</java.sql.
Timestamp>
        <byte>getByte(":column:")</byte>
        <java.sql.Array>getArray(":column:")</java.sql.Array>
        <java.lang.Object>getObject(":column:")</java.lang.Object>
        <UNKNOWN>getObject(":column:")</UNKNOWN>
    </dbfunctions>
    <Objectcasting>
        <sqlconverter>psqlConverter</sqlconverter>
        <datatypes>
            <point>
                <databasedriverdatatype>org.postgis.PGpoint<
/databasedriverdatatype>
                <customdatatype>psqlPoint</customdatatype>
                <datatype>piPoint</datatype>
            </point>
            <geometry>
                <databasedriverdatatype>org.postgis.PGgeometry<
/databasedriverdatatype>
                <customdatatype>psqlGeometry</customdatatype>
                <datatype>piShape</datatype>
            </geometry>
            <json>
                <databasedriverdatatype>json</databasedriverdatatype>
                <customdatatype>psqlJsonobject</customdatatype>
                <datatype>piJson</datatype>
            </json>
        </datatypes>
    </Objectcasting>
</language>
```

Most tags are self explanatory.

Exceptions are

`<tablenamefilter>` has a list of tags with characters that are not allowed in the program language. These are filtered out. For readability these tags are named <out></out>

<Objectcasting> these datatypes with their properties are programmed in the target project and defined by the programmers using this project.

Sourceroot map translation

The map structure is copied to the project source map.

Pseudo file naming is used to translate certain maps and filenames to database dependent naming.

The used pseudoname can be a part of a longer name.

Map and file names are processed equally.

pseudoname	translation
project	project name in lower case
Project	project name in lower case, first character Upper case
table	table name in lower case
Table	table name in lower case, first character Upper case
view	view name in lower case
View	view name in lower case, first character Upper case

All maps and filenames wich have the table or view pseudo filenaming will have a copy for each table or view found in the database.

Pseudo code translation

Pseudo code are all the database references used in the code templates.

All pseudo codes are enclosed in :, example :table:

Database driver dependent tags

These should be avoided, but in some cases can be needed

pseudoname	translation
:dbtool:	db tool name (see connectionsetup.xml)
:sqlconvertor:	convertor class/method for a custom defined data type

Non-database related tags

pseudoname	translation
:separator:	separator used in repeator tags
:metacoder_date:	date a file is generated, informative use
:,:	comma separator, is not repeated after a last item in a list
"//ProjectGenerator: NO AUTHOMATIC UPDATE"	a file containing this text will not be updated by the code generator
"//Custom code, do not change this line"	each block of code contained between this text (start and end) will be copied from the existing project code into the new created file

Pseudotag translation

this metacode is translated 1:1 with the active database reference

naming convention

(1) :tagname: - lower case

(2) :Tagname: - lower case, with first character Upper case

(3) :TAGNAME: - upper case

(4) :tagname_o: - original name as defined in database

metatags starting with repeat

repeater tags enclose a code block. That code block is repeated for each occurrence in the repeater.

A repeater should always be used in pairs

ex:

:repeattables:

:table:

:repeattables:

project

pseudoname	translation
:project:	(1)(2)(3)(4) project name

table - view

pseudoname	translation
:repeattables:	repeat all tables
:repeattables:	repeat all views
:repeattablesviews:	repeat all tables and views
:table:	(1)(2)(3)(4) table name
:view:	(1)(2) view name

primary key - foreign key - external foreign key - relational key

from the perspective of a table

external foreign key = a reference to this table from an external table

relational key = an external foreign where this table is part of the primary key in the relational table

pseudoname	translation
:repeatforeignkeys:	repeat foreign keys
:repeatuniqueforeignkeys:	repeat foreign keys, filter out double references to external tables
:repeatexternalforeignkeys:	repeat external foreign keys that are part of the primary key
:repeatallexternalforeignkeys:	repeat all external foreign keys, not limited to the primary key

:repeatuniqueexternalforeignkeys:	:repeatallexternalforeignkeys: without double references to external tables
:repeatreferences:	all distinct tables that have a reference as foreign keys from complete database, not part of a primary key (excluding doubles)
:uniquename:	(1)(2)(4) If a table reference is used only once in a table, that tablename is used. If not, a unique name is constructed for each foreign key referring to the same table.
:fkjavaname:	(1) foreign key name In this case, if there is double use of a table reference, a sequence number is added to the table name. :uniquename: is preferably used for readability
:pktable:	(1)(2)(4) primary key table - table reference of a foreign key
:pktablejavaname:	(1)(2) primary key table - table reference of a foreign key When there is double use of a primary key table, add a sequence number. This is usefull for naming variables in functions.
:pktablejavanamePK:	(1)(2) primary key table - table reference of a foreign key the suffix PK is added to the name. When there is double use of a primary key table, add a sequence number (after PK) This is usefull for naming variables in functions.
:extablename:	(1)(2)(4) used for naming foreign keys in external tables referring to this table
:exfkuniquename:	(1)(2) same as :uniquename: from the perspective of an external table
:reltablename:	(1)(2)(4) used for naming foreign keys in external tables referring to this table, with the condition the external table is a relational table and this table is referred to in the primary key.
:relfkuniquename: :relFkuniquename:	(1)(2) same as :uniquename: from the perspective of a relational table
:repeatforeignkeyfields:	repeat foreign key fields
:foreigncolumn:	(1)(2)(4) Foreign key column name
:primarycolumn:	(1)(2)(4) Foreign key column name in primary key table

column

pseudoname	translation
:repeatpkfields:	repeat primary key fields
:repeatfields:	repeat all fields not part of any key (pk, fk)
:repeatallfields:	repeat all fields

:column:	(1)(2)(3)(4) column name
:columnjavaname:	(1) name change in case a java restricted naming was used (ex. public) is only used in java code
:columntype:	programming language data type defined in program language settings files
:customfieldtype:	custom programmed data type defined in the project
:columntypesql:	SQL database data type
:columnsize:	column size - example varchar(**10**)
:getcolumndbfunction:	java.sql.ResultSet get functions for the column data type (in most cases getObject()
:columncast:	Databasedriver dependent java cast type Object (in case of custom objects). ex: GIS shapes
:columncustomtype:	custom data type, typical a custom object developed for this project
:columnposition:	db table column ordinal position
:counter:	counter value. For each counter tag, the value is increased by 1.

constraints

constrains are conditional metatags and are used inside a repeater

a conditional metatag encloses a code block

a conditional block ends up in the final code if the condition is valid, and can be combined with other constraints

pseudoname	translation
:iffieldtype:	inside a :iffieltype: block you can build in conditions for all java primitives and several data objects, in the form of their class name. These code blocks must each be enclosed in following tags • :boolean: • :byte: • :int: • :float: • :double: • :java.lang.String: • :java.sql.Date: • :java.sql.Time: • :java.sql.Timestamp: • :customfieldtype: (custom classes defined in project) • :Object: • :other: (all cases that are not specified in the iffieldtype block. All types above are optional. Any combination is allowed.
:isnotnullable:	column has constraint in database, can not be null
:inpk:	foreign key is part of primary key column is part of primary key
:notpk:	foreign key is not part of primary key column is not part of primary key
:infk:	column is part of a foreign key
:notfk:	column is not part of a foreign key
:infknotpk:	column is part of a foreign key and not part of the primary key

:inkey:	column is part of a key (pk, fk)
:notkey:	column is not part of a key (pk, fk)
:ifforeignkey:	used in :repeatexternalforeignkeys: to diffentiate between foreign keys and relational keys
:ifrelational:	used in :repeatexternalforeignkeys: to diffentiate between foreign keys and relational keys

extra

pseudoname	translation
:counter:	counter value. For each counter tag, the value is increased by 1.