# Support for H2 database

The HotRod H2 adapter automatically maps known database column types to DAO Java types. In most of the cases this default Java type is well suited to handle the database values. However, when needed the default Java type of a property can be overridden by a custom type if it's provided by the developer.

## Default Java Types

If a custom Java type is not specified HotRod will use the following rules to decide which Java type to use for each H2 column. In yellow is the DAO property type. In parenthesis the actual object type returned by the H2 JDBC driver, that on occasions may be different.

Please note that the Java types for the H2 columns may vary depending on the specific version and variant of the RDBMS, the operating system where the database engine is running, and the JDBC driver version.

|  |  |
| --- | --- |
| **H2 Column Type** | **Default Java Type** |
| TINYINT | java.lang.Byte |
| SMALLINT,  INT2,  YEAR | java.lang.Short |
| INTEGER,  INT,  MEDIUMINT,  INT4,  SIGNED,  IDENTITY | java.lang.Integer |
| BIGINT,  INT8 | java.lang.Long |
| DECIMAL(p,s),  DEC(p,s),  NUMERIC(p,s),  NUMBER(p,s) | If neither p or s are specified:   * java.math.BigInteger   If s is specified and different from zero the Java type is:   * java.math.BigDecimal   if s is not specified or specified with a value of zero:   * if p <= 2: java.lang.Byte * if 2 < p <= 4: java.lang.Short * if 4 < p <= 9: java.lang.Integer * if 8 < p <= 18: java.lang.Long * if p > 18: java.math.BigInteger |
| REAL,  FLOAT4 | java.lang.Float |
| DOUBLE,  DOUBLE PRECISION,  FLOAT,  FLOAT8 | java.lang.Double |
| CHAR(n),  CHARACTER(n),  NCHAR(n) | java.lang.String |
| VARCHAR(n),  LONGVARCHAR(n),  VARCHAR2(n),  NVARCHAR(n),  NVARCHAR2(n),  VARCHAR\_CASEINSENSITIVE(n),  VARCHAR\_INGNORECASE(n) | java.lang.String |
| CLOB(n),  NCLOB(n),  TINYTEXT(n),  TEXT(n),  MEDIUMTEXT(n),  LONGTEXT(n),  NTEXT(n) | java.lang.String |
| DATE | java.sql.Date |
| TIME | java.sql.Time |
| TIMESTAMP,  DATETIME,  SMALLDATETIME | java.sql.Timestamp |
| BINARY(n),  VARBINARY(n),  LONGVARBINARY(n),  RAW(n),  BYTEA(n),  BLOB(n),  TINYBLOB(n),  MEDIUMBLOB(n),  LONGBLOB(n),  IMAGE(n),  OID(n) | byte[] |
| BOOLEAN,  BIT,  BOOL | java.lang.Boolean |
| UUID | byte[] \* |
| ARRAY | No default HotRod data type |
| GEOMETRY | No default HotRod data type |
| OTHER | byte[] \*\* |

\* Even when H2's documentation states that UUID can be mapped to java.util.UUID this seems to work only when writing a value into H2. When reading the JDBC driver seems to produce a null value in all cases. A byte[] type, on the other hand, works consistently.

\*\* H2's documentation states that a java.lang.Object type can be used, but it does not work well in MyBatis. A byte[] type, on the other hand, works consistently.

## Custom Java Types

To override the default java type see the reference section for the tables, views, and selects. The example Custom DAO Property Java Types shows a some cases where a custom type overrides the default type. To do it add a <column> tag in a <table>, <view>, or <select> definition as in:

<table name=*"my\_table"*>

<column name=*"price"* java-type=*"java.lang.Double"* jdbc-type=*"NUMERIC"* />

</table>

This configuration will force the property type to java.lang.Double instead of java.math.BigDecimal (the default type).