JDBC Repository

Introduction

Generic Tables vs. Explicit

The JDBC repository supports two major approaches to mapping the OpenIDM objects to the database tables.

- Generic mapping: Allows arbitrary objects to be stored without special set-up or administration
- Explicit mapping: Allows for optimized storage and query by explicitly mapping a specific object type to the database

Each has its advantages and disadvantages.

The generic mapping allows for rapid development, and can make system evolution (and maintenance) simpler by providing a more stable DB structure. It comes at the cost of some overhead and not taking full advantage of DB facilities (validation outside the DB, less flexibility in indexing). Queries can also be trickier to set up initially.

The explicit mapping takes more effort to set up and maintain, but can fully leverage the native DB facilities.

Out of the box, well known tables used by the system such as for auditing utilize an explicit mapping. Tables for user definable objects use the generic mapping.

Configuration

Root object

Usage

```
{
    "dbType": string,
    "jndiName": string,
    "dbDriver": string,
    "dbUrl": string,
    "user": string,
    "password": string or crypto-object,
    "dbSchema": string,
    "maxBatchSize": number,
    "queries" : queries-object,
    "resourceMapping": resource-mapping-object
}
```

Properties

`"dbType"`: string, optional
The type of database to connect to. May affect the queries used and other optimizations.
Known supported types: MYSQL, DB2
If not specified, defaults to ANSI_SQL99

`"jndiName"`: string, optional
To use a data source in JNDI, set this property to the JNDI name of the data source.

Use only one of the 3 mechanisms to acquire a data source; either JNDI, JDBC driver manager, or OSGi services.

example: "jdbc/my-datasource"

`"dbDriver"`: string, optional
To use the JDBC driver manager to acquire a data source, set this property as well as dbUrl, user and password. Set dbDriver to the fully qualified class name of the database driver to use for your database.

Use only one of the 3 mechanisms to acquire a data source; either JNDI, JDBC driver manager, or OSGi services.

example: "com.mysql.jdbc.Driver"

`"dbUrl"`: string, optional
When using the driver manager, set to the driver specific URL to connect to your database.

example: "jdbc:mysql://localhost:3306/openidm"

`"user"`: string, optional
When using the driver manager, set to the user to use to connect to your database.

example: "openidm"

`"password"`: string or crypto-object, optional
When using the driver manager, set to the password to use to connect to your database.

Clear string passwords will automatically get encrypted by the system. To replace an existing encrypted (crypto-object) value, replace the whole value including brackets with a string of the new password.

example: "samplepwd"

`"dbSchema"`: string, optional
The database schema to use for openidm.

default: no schema prefix used for queries

example: "openidm"

`"maxBatchSize"`: number, optional
The maximum number of sql statements to batch together. Optimizes time to execute multiple queries. Some databases do not support batching, or limit how many statements can be batched. A value of 1 disables batching.

default: 1 (no batching)

example: 100

`"queries"`: queries-object, optional
Defines queries and gives them a name, which then can be referenced from configuration or code; e.g. for correlation queries. Supports tokens in the query for user provided arguments, or system variables.

`"resourceMapping"`: resource-mapping-object, optional
Defines the mapping between OpenIDM resource URIs (e.g. managed/user) and JDBC tables.

`"genericTables"`: map of generic-table-query-object, optional

Defines queries on generic tables.

queries-object

Usage

```
{
    "genericTables": { table-query-object, ...},
    "explicitTables": { table-query-object, ...}
}
```

Properties

`"genericTables"`: map of generic-table-query-object, optional
Defines queries to use on tables that use the generic db mapping.

`"explicitTables"`: map of generic-table-query-object, optional
Defines queries to use on tables that use the explicit db mapping.

table-query-object

Usage

```
"<query-name>" : query (string)
```

Properties

<query-name> (Property name): string, mandatory

Defines a unique name for the query

query (Property value): string, mandatory

Defines the SQL Query. Tokens can be inserted in the format

```
${<token-name}
```

Aside from the token syntax, the exact syntax of the query is dependent on the database used; as well as whether the table to query is an explicitly mapped table, or uses the generic mapping.

Example

```
"links-for-sourceId" : "SELECT * FROM ${_dbSchema}.${_table} WHERE sourceid = $
{sourceId}"
```

The tokens _dbSchema and _table are system provided variables.

The token sourceId is a user provided variable.

resource-mapping-object

Usage

```
{
    "default" : generic-table-mapping-value,
    "genericMapping": { generic-table-mapping-object, ...},
    "explicitMapping": { explicit-table-mapping-object, ...}
}
```

Properties

`"default"`: generic-table-mapping-object, optional
The default generic table; any resource that does not have a more specific mapping (either generic or explicit) will be stored in the default table.

`"genericMapping"`: map of generic-table-mapping-object, optional
Maps OpenIDM resources to tables structured to accommodate arbitrary objects, utilizing a generic table structure.

`"explicitMapping"`: map of explicit-table-mapping-object, optional
Explicitly maps OpenIDM resources to tables structured to accommodate specific objects.

generic-table-mapping-object

Usage

```
"<resource-uri-pattern>" : generic-table-mapping-value
```

Properties

`"<resource-uri-pattern>"`: string, mandatory

The URI pattern for the resources this mapping applies to. Can cover more than one resource

example: "managed/*"

generic-table-mapping-value

Usage

```
{
    "mainTable" : string,
    "propertiesTable": string,
    "searchableDefault": boolean,
    "properties" : generic-table-mapping-properties
}
```

Properties

`"mainTable"`: string, mandatory
The main table in generic table structure.

`"propertiesTable"`: string, mandatory
The properties table in generic table structure, used for searching.

`"searchableDefault"`: boolean, optional
Whether by default all properties of the resource should be searchable (and in turn stored and potentially indexed). Settings for individual properties can be overridden in the properties settings.

default: true

generic-table-mapping-properties

Usage

```
{
    "<property JsonPath>" : generic-table-mapping-property,     ...
}
```

Properties

`"<property JsonPath>"`: string, mandatory
The Json path pointing to a resource property.

example: "/firstname"

generic-table-mapping-property

Usage

```
{
    "searchable" : boolean
}
```

Properties

`"searchable"`: boolean, optional
Whether a property of a resource should be searchable (and in turn stored and potentially indexed). Overrides the default setting for this specific property.

Example

```
    "resourceMapping" : {
        "default" : {
            "mainTable" : "genericobjects",
            "propertiesTable" : "genericobjectproperties",
            "searchableDefault" : true
        },
        "genericMapping" : {
            "managed/*" : {
                "mainTable" : "managedobjects",
                "propertiesTable" : "managedobjectproperties",
                "searchableDefault" : true,
                "properties" : {
                    "/picture" : {
                        "searchable" : false
                    }
                }
            }
        },
        "explicitMapping" : {
            "audit/recon" : {
                "table" : "auditrecon",
                "objectToColumn" : {
                    "_id" : "objectid",
                    "action" : "activity",
                    "message" : "message",
                    "reconciling" : "reconciling",
                    "reconId" : "reconid",
                    "situation" : "situation",
                    "sourceObjectId" : "sourceobjectid",
                    "status" : "status",
                    "targetObjectId" : "targetobjectid",
                    "timestamp" : "activitydate"
                }
            }
        }
    }
```

explicit-table-mapping-object

Usage"<resource-uri>" : explicit-table-mapping-value

Properties

`"<resource-uri>"`: string, mandatory

The URI for the resources this mapping applies to.

example: "audit/recon"

explicit-table-mapping-value

Usage

```
{
    "table" : string,
    "objectToColumn": explicit-table-column-mapping
}
```

Properties

`"table"`: string, mandatory
The name of the db table to map to.

explicit-table-column-mapping

Usage

```
{
    "<property name>" : "<db column name>",
    ...
}
```

Properties

`"<property name>"`: string, mandatory
The name of the OpenIDM resource property.

example: "_id"

`"<db column name>"`: string, mandatory
The name of column to map to in the database table.

example: "openidmid"