Defining Connector Metadata

The connector metadata is an implementation of abstract java class **JDBCMetaData**. This defines what connection details need to be prompted to the user for creating a connection to a third-party source. This may include parameters like usernames, tokens, host names, ports, account names, etc.

Metadata Function List

The JDBCMetaData class is used for building a connection wizard for a data source. The following functions need to be implemented to create a basic connection wizard:

- public JDBCMetaData();
- public void initialiseParameters();
- public String buttonPressed(String buttonName) throws Exception;

These helper functions are also accessible in JDBCMetaData:

- protected final void addParameter(Parameter p);
- public void setParameterValue(String key, Object value);
- public final Object getParameterValue(String key);
- public boolean isParameterRequired(String key);
- public boolean hasDependentParameters(String key);

Metadata Function Definitions

public JDBCMetaData(); (Constructor)

The following attributes should be set in the constructor:

Attribute	Description
sourceName	Text name for the DataSource. For example, “Twitter Connector”.
sourceCode	A unique text code for the DataSource. For example, “TWITTER_CONNECTOR”.
driverName	The text class name of the DataSource. For example, “com.code.TwitterConnector”.
sourceType	This should always be DBType.THIRDPARTY.

Example implementation:

```
public SkiTeamMetaData() {

                        super();

                        sourceName = "Ski Team Source";
                        sourceCode = "SKI_DATA_SOURCE";
                        driverName = SkiTeamDataSource.class.getName();
                        sourceType = DBType.THIRD_PARTY;

}
```

top

public void initialiseParameters();

This function is where parameters should be registered. Registered parameters will be displayed to the user when creating a connection with this DataSource. Use the function addParameter() to add required parameters.

Example implementation:

```
public void initialiseParameters() {


                super.initialiseParameters();


                addParameter(new Parameter("HELP", "Connection Details", "Text", TYPE_NUMERIC,
DISPLAY_STATIC_TEXT, null, true));


                Parameter p = new Parameter("URL", "1. Request Access PIN", "Connect to twitter to receive a
PIN for data access",TYPE_UNKNOWN, DISPLAY_URLBUTTON, null,                 true);
                p.addOption("BUTTONTEXT", "Request URL");
                p.addOption("BUTTONURL", "http://google.com");
                addParameter(p);


                addParameter(new Parameter("PIN", "2. Enter PIN", "Enter the PIN recieved from Twitter",
TYPE_NUMERIC, DISPLAY_TEXT_MED, null, true));


                p = new Parameter("POSTPIN", "3. Validate Pin", "Validate the PIN", TYPE_TEXT, DISPLAY_BUTTON,
null, true);
                p.addOption("BUTTONTEXT", "Validate PIN");
                addParameter(p);


                addParameter(new Parameter("ACCESSTOKEN", "Access Token", "AccessToken that allows access to
the Twitter API", TYPE_TEXT, DISPLAY_PASSWORD, null, true));


                addParameter(new Parameter("ACCESSTOKENSECRET", "Access Token Secret", "AccessToken Password
that allows access to the Twitter API", TYPE_TEXT, DISPLAY_PASSWORD, null, true));


}
```

top

protected final void addParameter(Parameter p);

Parameter objects require the following metadata to be defined:

Attribute	Description
uniqueKey	Text Unique Key for this parameter.
displayName	Text description. This can be internationalised.
description	Parameter description. This can be internationalized.
defaultValue	Object to be assigned as the default value for this parameter.
displayType	DisplayType for this parameter. See DisplayType in appendix for more information.
dataType	DataType for this parameter. See Parameter DataType in appendix for more information

There are multiple constructors for Class Parameter, that allow for defining this object with a single line of Java code.

Some parameter display types require additional options, such as dropdown boxes and radio buttons. These need to be added to the parameter object after instantiation. For example:

```
Parameter p = new Parameter("URL", " Access PIN", "Connect to twitter to receive a PIN for data access",
TYPE_UNKNOWN, DISPLAY_URLBUTTON, null, true);

p.addOption("BUTTONTEXT", "Request URL");
p.addOption("BUTTONURL", "http://google.com");
addParameter(p);
```

top

---

public String buttonPressed(String buttonName) throws Exception;

This is a call-back for button UI elements. The function parameter buttonName holds the unique key for the button that called the callback function.

A button callback may be used to change the values of other parameters programmatically. A parameter can be set with *setParameterValue(String key, String value)*.

top

---

public void setParameterValue(String key, Object value);

Set the value of the parameter. Where function parameter key is the unique key of the parameter to be set and value is the value to assign to it.

top

---

public final Object getParameterValue(String key);

Get the value of a parameter. Where the function parameter key is the unique key of the parameter value to fetch.

top

---

public boolean isParameterRequired(String key);

To implement dependent filters the isParameterRequired() function can be overridden. Based on the values of other parameters, logic can determine whether the parameter with unique key should be shown.

For example:

```
public boolean isParameterRequired(String key) {

        if ("DOMAIN".equals(key)) {

                if ("SQL".equals(getParameterValue("WINDOWSAUTH"))) {

                        return false;

                }

        }

        return true;

}
```

top

---

public boolean hasDependentParameters(String key);

Return true for parameter with unique key if it has dependent parameters. This function is used to determine whether other parameter's visibility needs to be updated based on modification of this parameter's value.

top

Previous topic: Connector prerequisites
Next topic: Define data source