Eclipse DTP Project Component: Open Data Access Framework

Last updated: November 10, 2005

1. Introduction

This document provides an overview of the Open Data Access framework found in the DTP Milestone 1 (Iteration 2) build. It describes its intended usage, current functionalities, run-time API and extension point for community review and feedback.

1.1 What is Open Data Access (ODA)?

The Open Data Access (ODA) component is an open and flexible data access framework that provides a common way for a consumer application allows applications to access data from both standard and custom data sources. It enables data connectivity between data consumers and data source providers through published run-time interfaces (*org.eclipse.datatools.connectivity.oda*). The ODA framework is intended to provide an abstraction for scalable data retrieval from heterogeneous data sources. Thus, it serves a different purpose than other standard data-centric frameworks, such as JDBC and SDO, which all have significant focus on data updates among other things.

The ODA framework also includes an ODA driver management package (*org.eclipse.datatools.connectivity.oda.consumer.helper*) that helps an ODA consumer application to manage the diverse behavior and capabilities of individual ODA data drivers.

The DTP ODA framework is actively consumed by the Eclipse BIRT project.

An ODA data driver is created simply by implementing the run-time interfaces defined by the ODA framework. The run-time interfaces include support for establishing a connection, accessing meta-data, and executing queries to retrieve data. A driver can define internal data source connection profiles (in Milestone 1) and/or work with the Connection Management Framework's Connection Profiles extensions (in Milestone 2). Once developed, the driver can be registered through an extension point with individual ODA consumer components to enable data connectivity.

For an exemplary implementation of the DTP ODA run-time API, see the Flat File data source ODA driver (*org.eclipse.datatools.connectivity.oda.flatfile* project) in the DTP CVS repository. Additional DTP ODA data source plug-ins can be found in the Eclipse BIRT project source tree in CVS, and are available in the BIRT download builds: http://download.eclipse.org/birt/downloads .

2. ODA Run-time APIInterfaces

The ODA runtime interfaces define the primary run-time operations needed from an ODA data driver to access and retrieve data from a data source. Below is a brief overview of the main supported run-time operations. A custom ODA driver has a high-level of flexibility on the set of features it may want to support and implement.

Data Source Connection

- Establishes a live connection to any type of data source, based on data source-specific connection properties. The type of data source accessed is not restricted to RDBMS.

- Obtains information on the capabilities of a data source provider for each type of data set query.

- Creates one, or multiple, concurrent data set specific queries.

Data Set Query

- Prepares and executes a query text command. The query syntax is specific to an underlying data source, i.e. not restricted to SQL syntax.

- Assigns data set-specific runtime properties to a query.

- Handles one or more sets of data rows, i.e. result sets, from a single data set query.

- Allows access to a result set by name, if supported by an underlying data source.

Parameters

- Provides the run-time metadata of parameters specified in a prepared query.

- Handles scalar and complex input/output parameters, if supported by an underlying data source.

Result Sets

- Fetches each row of data columns in a sequential, forward direction from a result set.

- Allows sequential or concurrent access to multiple result sets, as supported by an underlying data source.

- Obtains the metadata of individual result set.

- Sets the limit on the maximum number of rows that can be retrieved by a query or a result set.

2.1 ODA Run-time API

The ODA runtime API interfaces are JDBC-like, but have been extended to support additional capabilities of non-RDBMS data sources. An ODA driver would implement the public runtime interfaces, which would in turn wrap data-source-specific APIs, such as web services, to retrieve a result set's data rows.

Below is a brief overview of the ODA run-time API's main interfaces:

- The *IDriver* serves as the entry point to an ODA runtime driver. The driver produces a dedicated *IConnection* object for establishing a connection to the underlying data source provider.

- An open connection in turn creates an *IQuery* to define the specifics of a data set query.

- A connection also provides meta-data information on the capabilities of the data source, and its supported data set types in *IDataSetMetaData*.

- A query is executed to retrieve one or more *IResultSet* instances.

- A result set is then used to fetch result data. It also provides an *IResultSetMetaData* for its meta-data information.

For detail ODA API documentation, see its Javadoc API Reference Documentation included in the DTP download.

2.1.1 ODA Data Types

The kind of data types supported by the ODA run-time API are listed below:

- String
- Integer
- Double
- BigDecimal
- Date – includes time portion, up to seconds
- Time – no date portion
- Timestamp – similar to Date, and includes up to nano-seconds
- Blob
- Clob
- Structure – a single row of scalar data fields
- Table – zero or more homogenous rows of scalar data fields

For detail ODA API documentation, see its Javadoc API Reference Documentation included in the DTP download.

3. ODA Run-time Extension Point

Identifier:

org.eclipse.datatools.connectivity.oda.dataSource

Description:

This extension point is used to support the extension of design-time and run-time data source access by a data application. Each extension must implement the Open Data Access (ODA) Java runtime interfaces defined in the *org.eclipse.datatools.connectivity.oda* package.

Configuration Markup:

<!ELEMENT extension (dataSource , dataSet+)>

<!ATTLIST extension

point CDATA #REQUIRED

id CDATA #IMPLIED

name CDATA #IMPLIED>

<!ELEMENT dataSource (traceLogging? , properties?)>

<!ATTLIST dataSource

id CDATA #REQUIRED

odaVersion CDATA "3.0"

defaultDisplayName CDATA #IMPLIED

driverClass CDATA #REQUIRED

setThreadContextClassLoader (true | false) "false">

The definition of a type of ODA data source extension for use at design-time and run-time.

- **id** - A fully qualified ID that uniquely identifies this ODA data source extension within an ODA consumer application's environment. If a data source designer extension (TBD) for this ODA driver is available, the value of this attribute must match that of the designer extension's *id* attribute in its data source element.

- **odaVersion** - Version of the ODA interfaces for which this driver is developed. This element is required and should take the format of Major.Minor or Major.Minor.Service (e.g. 3.0 or 2.0.1).

- **defaultDisplayName** - The display name of the ODA data source extension. Its value can be localized by using the plugin.properties mechanism. Default to the extension id if no display name is specified. It can be used by an ODA consumer application's designer tool in displaying a list of ODA data source extensions, when they do not have a corresponding data source editor (data source UI extension point).

- **driverClass** - Concrete class that implements the *org.eclipse.datatools.connectivity.oda.IDriver* interface. This is the entry point of the ODA runtime driver. The same driver may support multiple data source extensions.

- **setThreadContextClassLoader** - If true, the consumer of the ODA runtime extension plug-in should set the thread context class loader to the one used to load this driver before calling any ODA interface method. Any data source plug-in extension with this flag set to true would take precedence, and is applied to all data source extensions implemented by this plug-in.
If the thread context class loader being set is the OSGi class loader that was used to load this ODA runtime plugin, it is not designed to be used by a plugin to in turn load additional classes. If further class loading is needed, it is up to individual ODA runtime plugin implementation to provide its own URLClassLoader, and switch thread context class loader as appropriate.

<!ELEMENT dataSet (dataTypeMapping+ , properties?)>

<!ATTLIST dataSet

id CDATA #REQUIRED

defaultDisplayName CDATA #IMPLIED>

The definition of a type of data set supported by the dataSource extension.

- **id** - A fully qualified ID that uniquely identifies this ODA data set definition within an ODA consumer application's environment.

- **defaultDisplayName** - The display name of the ODA data set definition. Its value can be localized by using the plugin.properties mechanism. Default to its id if no display name is specified.

<!ELEMENT dataTypeMapping (alternativeOdaDataType*)>

<!ATTLIST dataTypeMapping

nativeDataType CDATA #IMPLIED

nativeDataTypeCode CDATA #REQUIRED

odaScalarDataType (Date|Double|Integer|String|Time|Timestamp|Decimal|Blob|Clob) "String">

A data types mapping from a data provider's native data type to one or more ODA data types. Each native data type must be mapped to a primary ODA scalar data type. The driver can optionally provide a list of alternate ODA data types to which it is capable of converting a native data type. This data type mapping facilitates all ODA consumers to map from the same set of ODA data types to its own application-specific data types.

- **nativeDataType** - Native data type name (a string). Used for information only.

- **nativeDataTypeCode** - Native data type code (an integer). Its value must match one of the data type codes returned in the driver's ODA interface implementation.

- **odaScalarDataType** - The primary ODA scalar data type which the native type maps to. Supported ODA data types are: Date, Double, Integer, String, Time, Timestamp, Decimal, Blob and Clob.

<!ELEMENT alternativeOdaDataType EMPTY>

<!ATTLIST alternativeOdaDataType

odaScalarDataType (Date|Double|Integer|String|Time|Timestamp|Decimal|Blob|Clob) >

Provide an alternative mapping to an ODA scalar data type.

- **odaScalarDataType** - The ODA scalar data type to which the native type may be converted by the driver.

<!ELEMENT traceLogging EMPTY>

<!ATTLIST traceLogging

logLevel CDATA "WARNING"

logFileNamePrefix CDATA #IMPLIED

logDirectory CDATA #IMPLIED

logFormatterClass CDATA #IMPLIED>

Configures the ODA run-time driver's trace logging settings for the data source extension. The configured values are passed through to the driver's implementation of the `IDriver.setLogConfiguration` method.
It is up to individual ODA driver on how to honor any of these trace logging attributes as appropriate.

Note: The trace logging configuration specified in the plug-in PDE .options file would take precedence over those configured in this element, if the debug tracing flag is set to "true".
The ODA plug-in's PDE tracing options, listed below for cross reference, match the attributes of this element.
<plug-in Id>/debug = true/false
<plug-in Id>/traceLogging/logLevel
<plug-in Id>/traceLogging/logFileNamePrefix
<plug-in Id>/traceLogging/logDirectory
<plug-in Id>/traceLogging/logFormatterClass

- **logLevel** - The name or numeric value for the driver's log level.
The log levels' names and corresponding numeric values are:
"SEVERE" = 1000; "WARNING" = 900; "INFO" = 800; "CONFIG" = 700; "FINE" = 500; "FINER" = 400; "FINEST" = 300; "ALL" = 0; "OFF" = 1001 or higher .

- **logFileNamePrefix** - A string prefix for driver's log file name.

- **logDirectory** - Directory for log file.

- **logFormatterClass** - The class name of a concrete log formatter, suitable for use by the driver-

specific logging utility.

<!ELEMENT property (choice*)>

<!ATTLIST property

name CDATA #REQUIRED

defaultDisplayName CDATA #IMPLIED

type (string|choice) "string"

canInherit (true | false) "true"

defaultValue CDATA #IMPLIED

isEncryptable (true | false) "false">

A property whose value can be edited at design-time using an ODA consumer application's designer tool. Its value is then passed to the ODA runtime driver during run-time.

- **name** - Unique name of the property.
- **defaultDisplayName** - The default display name. Its value can be localized by using the plugin.properties mechanism.
- **type** - Type of the property. The property type could be one of the values listed in the Restriction enumerations.
- **canInherit** - Reserved.
- **defaultValue** - Default value of the property, if no property value is set.
- **isEncryptable** - A flag indicating whether this property value is encryptable. Setting it to "true" indicates to an ODA consumer application that this property's value should be encrypted.

<!ELEMENT propertyGroup (property+)>

<!ATTLIST propertyGroup

name CDATA #REQUIRED

defaultDisplayName CDATA #IMPLIED>

A grouping of one or more properties in an ODA consumer application's designer tool. The group attributes are for display only. All properties listed under a propertyGroup are handled as scalar properties at run-time.

- **name** -

- **defaultDisplayName** - The default display name. Its value can be localized by using the plugin.properties mechanism.

<!ELEMENT propertyVisibility EMPTY>

<!ATTLIST propertyVisibility

name CDATA #REQUIRED

visibility (change|lock|hide) >

Used to set the visibility level of the named property when it is shown in the property sheet of an ODA consumer application's designer tool.

- **name** - The name of a property that is defined either by this data source extension or is a system-defined property.
- **visibility** - The valid options are: change, hide, lock.

<!ELEMENT choice EMPTY>

<!ATTLIST choice

name CDATA #REQUIRED

value CDATA #IMPLIED

defaultDisplayName CDATA #IMPLIED>

Choice of property values.

- **name** - Name of the choice
- **value** - Value to be used, if the given choice is selected.
- **defaultDisplayName** - The default display name. Its value can be localized by using the plugin.properties mechanism.

<!ELEMENT properties (property* , propertyGroup* , propertyVisibility*)>

A collection of property definitions in a data source extension or its supported data set definitions.

API Information:

The data source extension's driver must implement the interfaces defined in the

org.eclipse.datatools.connectivity.oda package. See the package's JavaDoc documentation and API interfaces for more information.

Supplied Implementation:

The plugin *org.eclipse.datatools.connectivity.oda.flatfile*, supplied with the Eclipse DTP Connectivity source, provides an example for implementing a simple ODA run-time extension.
In addition, a set of default implementation of the main ODA runtime interfaces are provided in the *org.eclipse.datatools.connectivity.oda.impl* package. These classes assume the behavior of a simple ODA driver, and have labeled TODO tasks for a driver developer to implement data source specific behavior. These concrete classes are provided for use as templates or base classes to aid in the development of an ODA custom driver. Their use is purely optional.