New Journey of Web Approach:O-Data

Shravani Vanka, Sharon Leo F, Bipradip Roy
Dept. of Computer Applications,
Dayananda Sagar College of Arts, Science and Commerce,
Bangalore, India.

***Abstract*— There are quite a lot of interest showing how to create OData services using Web API OData, but these requires Entity Framework and a database server behind. If you want a quick try or you have your own way of implementing data sources, these tutorials may not be the best fit. In this paper, we will show how to build an OData service using in-memory data as data source and with basic function.**

Keywords—Odata, Http, API.

I. DESCRIBING ODATA

Our world is awash in data. Vast amounts exist today, and more is created every year. Yet data has value only if it can be used, and it can be used only if it can be accessed by applications and the people who use them. Allowing this kind of broad access to data is the goal of the Open Data Protocol, commonly called just OData. OData (Open Data Protocol) is an OASIS standard that defines the best practice for building and consuming RESTful APIs. OData helps you focus on your business logic while building RESTful APIs without having to worry about the approaches to define request and response headers, status codes, HTTP methods, URL conventions, media types, payload formats and query options etc. OData also guides you about tracking changes, defining functions/actions for reusable procedures and sending asynchronous/batch requests etc. Additionally, OData provides facility for extension to fulfill any custom needs of your RESTful APIs.

II. THE PROBLEM: ACCESSING DIVERSE DATA IN A COMMON WAY

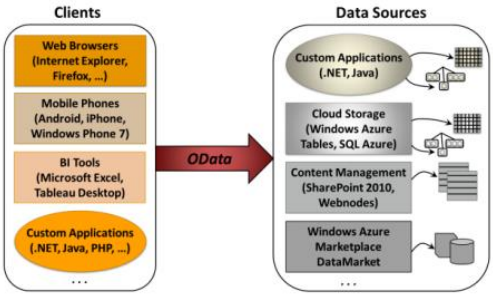
There are many possible sources of data. Applications collect and maintain information in databases, organizations store data in the cloud, and many firms make a business out of selling data. And just as there are many data sources, there are many possible clients: Web browsers, apps on mobile devices, business intelligence (BI) tools, and more. How can this varied set of clients access these diverse data sources?

One solution is for every data source to define its own approach to exposing data. While this would work, it leads to some ugly problems. First, it requires every client to contain unique code for each data source it will access, a burden for the people who write those clients. Just as important, it requires the creators of each data source to specify and implement their own approach to getting at their data, making each one reinvent this wheel. And with custom solutions on both sides, there's no way to create an effective set of tools to make life easier for the people who build clients and data sources.

Thinking about some typical problems illustrates why this approach isn't the best solution. Suppose a Web application wishes to expose its data to apps on mobile phones, for instance. Without some common way to do this, the Web application must implement its own idiosyncratic approach, forcing every client app developer that needs its data to support this. Or think about the need to connect various BI tools with different data sources to answer business questions. If every data source exposes data in a different way, analyzing that data with various tools is hard -- an analyst can only hope that her favorite tool supports the data access mechanism she needs to get at a particular data source.

III. THE SOLUTION: WHAT ODATA PROVIDES:

OData defines an abstract data model and a protocol that let any client access information exposed by any data source. Diagram shows some of the most important examples of clients and data sources, illustrating where OData fits in the picture.



Any OData client can access data provided by any OData data source.As the figure illustrates, OData allows mixing and matching clients and data sources. Some of the most important examples of data sources that support OData today are:Custom applications: Rather than creating its own mechanism to expose data, an application can instead use OData. Facebook, Netflix, and eBay all expose some of their information via OData today, as do a number of custom enterprise applications. To make this easier to do, OData libraries are available that let .NET Framework and Java applications act as data sources.Cloud storage: OData is the built-in data access protocol for tables in Microsoft's Windows Azure, and it's supported for access to relational data in SQL Azure as well. Using available OData libraries, it's also possible to expose data from other cloud platforms, such as Amazon Web Services.Content management software: For example, SharePoint 2010 and Webnodes both have built-in support for exposing information through OData.Windows Azure Marketplace DataMarket: This cloud-based service for discovering, purchasing, and accessing commercially available datasets lets applications access those datasets through OData.While it's possible to access an OData data source from an ordinary browser -- the protocol is

based on HTTP -- client applications usually rely on a client library. As above fig. shows, the options supported today include:Web browsers: JavaScript code running inside any popular Web browser, such as Internet Explorer or Firefox, can access an OData data source. An OData client library is available for Silverlight applications as well, and other rich Internet applications can also act as OData clients.Mobile phones.

IV. ODATA QUERY STRING OPTIONS

The Query Options section of an OData URI specifies three types of information: System Query Options, Custom Query Options, and Service Operation Parameters. All OData services must follow the query string parsing and construction rules defined in this section and its subsections.

Option	Description
$expand	Directs that related records should be retrieved in the record or collection being retrieved.
$filter	Specifies an expression or function that must evaluate to true for a record to be returned in the collection.
$orderby	Determines what values are used to order a collection of records.
$select	Specifies a subset of properties to return.
$skip	Sets the number of records to skip before it retrieves records in a collection.
$top	Determines the maximum number of records to return.

$expand :

Directs that related records should be retrieved in the record or collection being retrieved. If you want to retrieve related records, locate the name of the entity relationship that defines this relationship. You may have to view the entity relationship information in the application to correctly identify the relationship or the conceptual schema definition language (CSDL) for the Organization Data Service.

For example:

To retrieve opportunity records related to accounts, use theopportunity_customer_accounts entity relationship. The query /AccountSet?$expand=opportunity_customer_accounts returns the opportunity records and the account records.

If you're limiting the columns returned, you must also include the name of the navigation property in the query. For example, the query /AccountSet?$select=Name,opportunity_customer_accounts&$expand=opportunity_customer_accountsreturns only the account name and the expanded opportunity records.

$filter

Specifies an expression or function that must evaluate to **true** for a record to be returned in the collection.If you're retrieving additional sets of data using the **next** link, you shouldn't change the **$filter** query option value because this causes unpredictable results. The OData specification for the Filter System Query Option describes the operators used to create an expression to evaluate in the filter. Microsoft Dynamics CRM uses all the logical operators and a subset of the functions available, but doesn't support use of the arithmetic operators.

V. USING DIVERSE DATA SOURCES WITH DIFFERENT BI TOOLS:

Business intelligence, analyzing information to extract meaning, is an important part of how people use data. Analyzing data first requires accessing data, and given the multiplicity of BI tools and data sources in use today, this is a non-trivial problem. Different analysts prefer different tools, and data is kept in different forms in different places. Much of an organization's useful data is likely to be wrapped inside custom and packaged applications, for example, while many organizations also keep useful business data in SharePoint lists. Another possible source for data is Microsoft's Windows Azure Marketplace DataMarket, which provides a cloud-based way to purchase and access commercial data sets.Suppose an analyst wishes to combine data from these various sources. Maybe a retailer is trying to decide where to locate a new store, for example, and so needs to look at sales information from one of its custom applications, customer survey data stored in SharePoint lists, and demographic data acquired from DataMarket. Or perhaps analysts in a local government wish to access emergency call data from the city's custom call center application, police reports stored in SharePoint, and national crime statistics available through DataMarket. In both cases, it's entirely possible that different analysts wish to use different tools to work with this data.The problem is clear: How can we connect multiple clients to multiple data sources? Without a common approach to exposing and accessing data, the situation is bleak. OData can help.

Different BI tools can use OData to access data stored in different formats across different data sources.In this example, two different analysts using different BI tools -- Tableau Desktop and Microsoft Excel's PowerPivot -- are accessing data from the three data sources just listed : SharePoint 2010 lists, a custom application, and Windows Azure Marketplace DataMarket. All of these technologies can use OData today, and so making these connections is straightforward. Because clients and data sources speak the common language of OData, hooking them together gets simpler, and analysts can begin working with new data more rapidly.

Examining OData: A Closer Look at the Technology and Its Implementation

OData began life as a Microsoft project code-named Astoria. The technology was then renamed ADO.NET Data Services before its protocol and data model were separated out and became OData. (The parts of ADO.NET Data Services that were focused on the Windows implementation of OData are now known as WCF Data Services.) Whatever the name, though, the fundamental technology of OData has remained the same.As described earlier, it's useful to think about the OData world in four parts: the data model, the protocol, the client libraries, and the OData service itself. This section describes all four, beginning with the data model.

VI. RELATIONAL OPERATORS

OData protocol enables you to use other relational operators such as not equal (ne), less than (lt), less or equal (le), greater than (gt), greater or equal (ge). Some examples are:
/Products?$filter=ID lt 4
/Products?$filter=ID ge 3
/Products?$filter=ID ne 2
Logical operators
You can create complex queries using the logical operators and, or, not and brackets. One example of complex query is shown in the following example:
/Products?$filter=ID lt 7 and (Name eq 'Milk' or ID gt 3) or not(ID le 4 and ID ge 6)
Arithmetical operations
You can apply standard operators to add (add), subtract (sub), multiply (mul), divide(div), or find remainder(mod). Example of query that returns all products where a total value in stock (unit price * units in stock) is less that 45 (with condition that there are some items in stock) is:/Products?$filter=(UnitPrice mul UnitsInStock) lt 45 and UnitsInStock ne 0.Note that currently you can use arithmetical functions only in $filter condition but not in the $select.

Numerical functions
If your properties are numbers you can apply floor, ceiling, and round functions. Example of the query that uses these functions is /Products?$filter=floor(Price) eq 3 or ceiling(Price) eq 3
String functions
There are a lot of string functions you can use in your filter expressions - some of them (with examples) are:
length(string) /Customers?$filter=length(CompanyName) lt 10
trim(string), toupper(string), tolower(string) - /Products?$filter=toupper(Name) eq 'MILK'
substringof(‘part’, text) /Customers?$filter=substringof(‘Sales’,ContactTitle)
endswith(text, ‘part’) /Customers?$filter=endswith(ContactTitle,'Manager')
startswith(text, ‘part’) /Customers?$filter=startswith(ContactTitle,'Sales')
Date functions
If you have datetime properties in the resources you can use several date part functions such as year(), month(), day(), hour(), minute(), and second(). As an example, URL query that returns all orders with OrderDate in 1996 is:
/Orders?$filter=year(OrderDate) eq 1996
Ordering results
Another usable feature is ordering. You can order results by some property or function using the $orderby query option. Default order is ascending but you can change it. Some examples are:
/Products?$orderby=ID desc
/Products?$orderby=length(Name)
Updating data
OData services are REST services so data modification operations are also allowed. In the REST services a type of data access operation is defined using the type of HTTP request that is sent using the following rules:
HTTP GET request is used to read data
HTTP POST request is used to update existing entity
.

VII. CONCLUSION

Our world really is awash in data. Yet too much of it is locked up in silos, inaccessible to many of the applications that might use it. By providing a common way for diverse clients to access an array of data sources, OData can help set this information free.Because it relies on a simple abstract data model based on entities and associations, OData can be used with many kinds of data. Because it builds on familiar technologies such as REST, Atom/AtomPub, and JSON, OData isn't especially hard to understand. And because support is available for creating clients and services on various platforms and devices, OData is straightforward to implement.
The value of a common approach to accessing data is undeniable. Reflecting this, many clients and data sources support OData today. Going forward, expect to see that support get broader still. Our data is just too valuable to keep locked away.

VIII. REFERENCES

1. http://www.odata.org/getting-started/basic-tutorial/
2. http://www.odata.org/getting-started/
3. http://www.odata.org/getting-started/advanced-tutorial/
4. http://www.codeproject.com/Articles/393623/OData-Services
5. http://www.asp.net/web-api/overview/odata-support-in-aspnet-web-api/odata-v4/create-an-odata-v4-endpoint
6. https://msdn.microsoft.com/en-us/data/gg601462.aspx
7. http://www.asp.net/web-api/overview/odata-support-in-aspnet-web-api
8. https://github.com/ODataOrg/tutorials
9. https://olingo.apache.org/doc/odata4/tutorials/read/tutorial_read.html