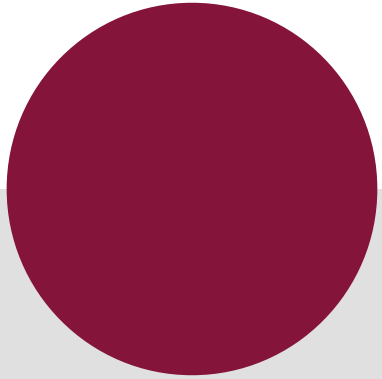
.msg .consulting .solutions .partnership



Intro to OData

Background

- Once REST APIs became prevalent, another problem popped out: **standardization**.
- Basically, each software that exposes REST follows its own conventions.
- There are several attempts to build a standard for consuming / discovering REST API structures:
  - HATEOAS
  - JSON-API
  - OData
- The goal is to build something that can be consumed easily, without the need of prior knowledge about the structure of the API.

Mission

- OData (Open Data Protocol) is an ISO/IEC approved, OASIS standard that defines a set of best practices for building and consuming RESTful APIs.

- OData helps you focus on your business logic while building RESTful APIs without having to worry about the various approaches to define request and response headers, status codes, HTTP methods, URL conventions, media types, payload formats, query options, etc.

- OData also provides guidance for tracking changes, defining functions/actions for reusable procedures, and sending asynchronous/batch requests.

- OData RESTful APIs are easy to consume. The OData metadata, a machine-readable description of the data model of the APIs, enables the creation of powerful generic client proxies and tools.

Backing Companies

- Microsoft: authored the first version of the protocol back in 2007.

- Then a committee was created for discussing improvements and changes:
  - Microsoft
  - SAP
  - IBM
  - Red Hat
  - Citrix
  - And more…

Criticism

- OData exposes a structure notably similar to a persistence data model.
- As such, a lot of OData implementations are simply “database browsers”.
- Exposing the persisted data model via a HTTP interface is an architectural anti-pattern.

Versions

- V1: Used mostly by Microsoft, was valuable to extract usage scenarios and much needed feature requests.
- V2: By far the most popular for the most time. Is still used today.
- V3: Mostly just a bridge version with some small improvements.
- V4: The current standard, is being pushed heavily by SAP (but not yet 100% supported in all technologies).

Libraries

- There are several libraries that can produce or consume OData.

- In msg, we use:
  - [ABAP; Producer] SAP Gateway
  - [Java; Producer and Consumer] Apache Olingo
  - [Java; Producer and Consumer] SAP Cloud SDK (a wrapper for Olingo)
  - [JavaScript; Consumer] SAP UI5

- Check out https://www.odata.org/ecosystem/ for more libraries.

Structure

- For the example OData service: https://services.odata.org/V2/Northwind/Northwind.svc/
- https://services.odata.org/V2/Northwind/Northwind.svc/ is the service root (lists all endpoints).
- https://services.odata.org/V2/Northwind/Northwind.svc/$metadata is the metadata document, describing the service structure, data types, etc.

Metadata

- https://services.odata.org/V2/Northwind/Northwind.svc/$metadata
- Simple Types = built-in types like Edm.Int32.
- Complex Types = user-defined structures consisting of several simple fields (e.g. Address).
- Entity Types = user-defined structures consisting of simple or complex fields (e.g. Order).
- Navigations = relationships between entities.
- Entity Sets = collections of entities of a given Entity Type. Each entity set has it's own endpoint.
- Function Import = custom operation which may accept scalar parameters and may return one or more entities, complex structures or simple values.

Operations

- For example let's take the Orders entity set.

- Read: GET /Orders(1)
- Update: PUT /Orders(1)
- Delete: DELETE /Orders(1)
- Create: POST /Orders
- Query: GET /Orders?$filter=…&$top=…&$skip=…

- Other important parameters / endpoints:
  - $format for controlling the data format (JSON vs XML).
  - $count / $inlinecount for getting back the total count of entities.
  - $expand for "expanding" related entities (and return them in a single response).
  - $batch for executing multiple OData operations in a single HTTP request.

Components

- In all producers, there are usually two major components:
  - The “model provider”: MPC in SAP Gateway, EDM Provider in Olingo.
  - The “data provider”: DPC in SAP Gateway, the Single Processor in Olingo.

- In Olingo:
  - OData servlet: entry point for the framework.
  - ODataServiceFactory: a factory class that the user must extend to connect the model and the data providers.
  - EdmProvider: interface of the model provider.
  - ODataSingleProcessor: interface of the data provider, has methods for handling each type of operation (validation based on the data model is done by the framework before).
  - ODataErrorCallback: interface for handling exceptions (e.g. to add logging; the error response is generated automatically).

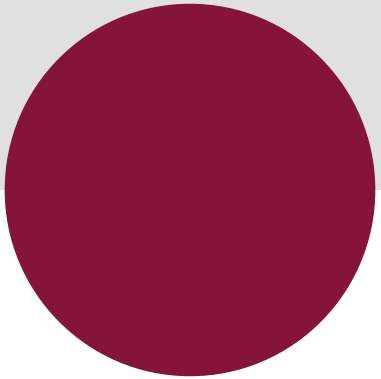
Olingo libraries

- “Core”: users must manually define both the EdmProvider and the single processor.
- “JPA”: everything is automatically generated based on the JPA entity model. Problem: harder to add business logic in between the HTTP endpoint and the DB.
- “Annotation”: Edm is deduced via annotations and simple data conversion is done; instead of the single processor, you need to implement a Data Store.
- SAP SDK: based on Olingo; data model is generated from a metadata document, single processor is based on annotated methods.

msg systems ag
Robert-Buerkle-Str. 1, 85737 Ismaning/Munich
Germany

Phone: +49 89 96101-0
Fax: +49 89 96101-1113
info@msg.group

www.msg.group



.msg .consulting .solutions .partnership