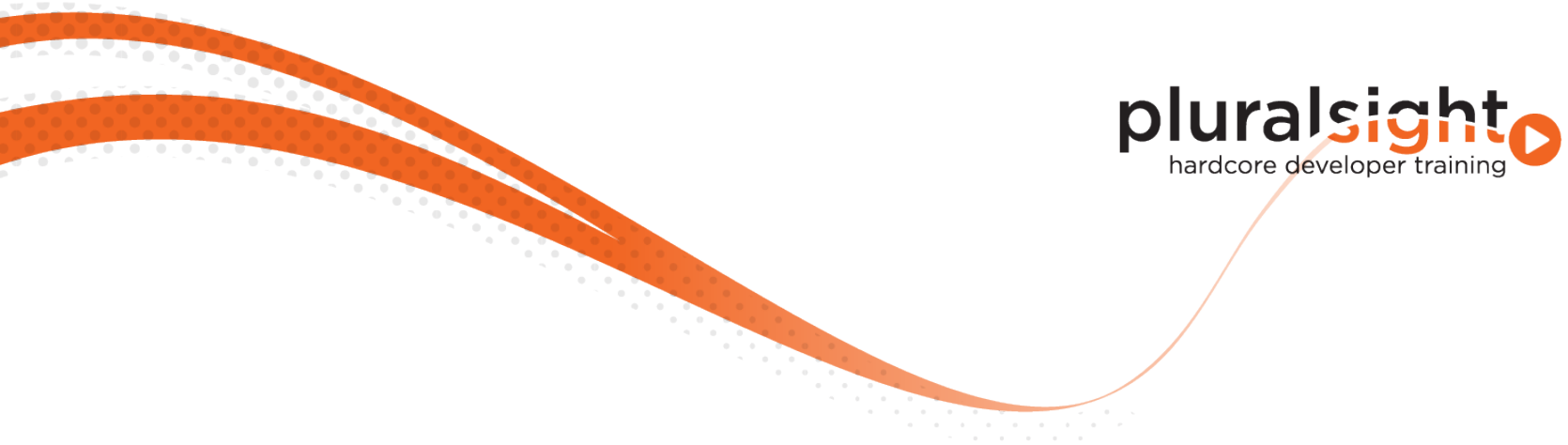
Updating Web API Resources Through OData

Brian Noyes
CTO, Solliance (www.solliance.net)
@briannoyes



pluralsight
hardcore developer training

Outline

- Exposing update operations from EntitySetControllers
- Returning OData compliant errors
- Updating relations through OData
- OData Actions and Unhandled Requests

Updating Resources through OData

- **OData follows the conventions of HTTP and REST**
  - POST => Create (Add/Insert)
  - PUT => Update
  - DELETE => Delete
- **OData adds support for PATCH**
  - Partial update
  - Only send the property values that have changed
    - Key-value pairs
  - Merge with the other properties on the back end
  - More efficient at the wire level and when executing against the DB
- **Payloads are encoded using OData ATOM or JSON**
- **Addressing scheme is any OData path that addresses a single resource**
  - Web API out of the box will only let you address the entities for the current controller
    - CustomersController to update Customers, OrdersController to update Orders, etc.

Adding Update Operations to EntitySetController

- **Base class defines update APIs**
  - Handles the direct POST / PUT / DELETE / PATCH calls
  - Calls virtual methods for CreateEntity, UpdateEntity, PatchEntity
    - Parameters get handled by the base class HTTP verb handlers, passed to the virtual methods strongly typed
  - Handle the DELETE by overriding the Delete method on base
    - Need to use [FromODataUri] attribute on string key parameters
      - To remove the quotes
  - Handle “Prefer” header
    - Indicates desire for returned content on PUT and POST

Returning OData Compliant Errors

- **Use HTTP Status codes like a normal HTTP / REST service**
- **Return an OData protocol compliant error in the payload**
  - Error containing
    - Code – string
    - Message – string
    - InnerError – nested error

Updating Relations Through OData

- **Not all relations are expressed through foreign key properties on your EDM**
  - Order may have a Customer property, but not a CustomerID property
- **Using foreign key properties on the objects is not enough in the general sense**
  - But when foreign key properties do exist, relations can be updated through a simple property update on the entity
- **OData defines separate scheme for updating links:**
  - $links addressing - /odata/Orders(12345)/$links/Customer
  - url element in the message body
    - ATOM or JSON
  - Use appropriate HTTP verb (POST, PUT, DELETE) for the kind of modification (Add, Update, Delete link)

OData Actions

- **Allows you to add additional action methods to your controllers that don't map to an explicit CRUD operation on a resource**
  - Start a workflow
  - Modify several related resources in a single action
  - Execute business logic
  - Perform validation
- **Must correspond to a single resource**
  - i.e. /Customers('ALFKI')
- **Modify EDM model to include action**
  - Name
  - Parameters
  - Return type
- **Add method to controller that takes the key and an ODataActionParameters**

Handling Unmapped Requests

- **EntitySetController has a HandleUnmappedRequest method**
  - Takes in an ODataPath parameter
  - Will be called for any request that mapped to that controller but did no handling method found on the controller
  - Good for debugging / tracing
  - Can use it to figure out when requests are getting to the right controller but you have something mismatched in the request URI segments and the method declarations

Summary

- OData can be used for updating resources as well
- Follows HTTP verb conventions of REST
- Payloads encoded as ATOM or JSON same as queries
- Need to return OData compliant errors
- Can update relations through link create / update / delete calls with appropriate verbs
- Can add Actions to resources and handle them in additional POST handlers in your controllers
- Can intercept unhandled requests to your controllers for debugging and diagnostic purposes