Open Services for Lifecycle Collaboration
open community. open interfaces. open possibilities.

Designing a core specification for OSLC

February 24, 2010
Dave Johnson - IBM / Rational

Disclaimer: The core specification described herein is a proposal & does not have OSLC approval.

The need for a core specification

- OSLC Work Groups (WGs) are:
  - Creating their own RESTful protocols
  - Designing their own XML, RDF/XML and JSON representations
  - Inventing new patterns for creating and managing resources
  - Completing and “converging” 1.0 specs
  - Planning and designing 2.0 specs
- Those are mostly good things
  - We’ve learned a lot in the process of developing some solid 1.0 specs
  - Good progress on 2.0 thinking around query, resource shapes, links
- Now we need to:
  - Ensure consistency and architectural integrity across specs
  - Build a core spec that defines our REST protocol and representations
  - Enable WGs to focus on data model and domain specific operations

Workgroup pain points

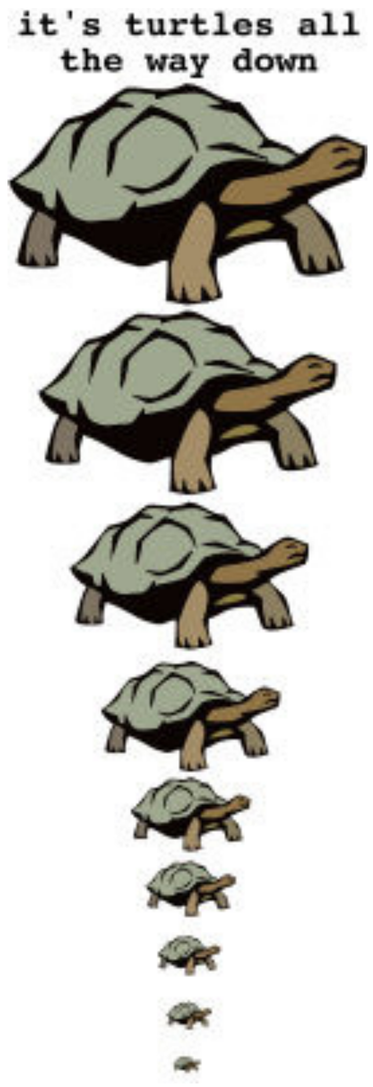
- Writing too much boiler-plate REST API spec text
- Have to know too much about RDF/XML, Atom, JSON
  - Too much discussion of representation
- Have to design and specify (or cut-and-paste)
  - RDF, JSON and other representations
  - Query syntax, semantics
  - Service documents
  - Delegated UI
- Have to invent new patterns for things like:
  - Resource Shapes
  - Modeling Links
  - Partial Update
  - File and File Descriptor
  - Hierarchical Web Content
  - Resources with huge numbers of properties

OSLC core spec design goals

- Stay true to the WWW and REST
  - e.g. focus on resources, uniform interface, stable/opaque URIs
- Be as RDF friendly as possible
  - e.g. focus on properties, provide RDF representations
- Balance tension between consistency & flexibility
  - Want consistency but don’t want to unduly constrain innovation
- Keep it simple
  - e.g. minimize new concepts introduced & specifications referenced
- Yet still manage to please everybody
  - e.g. schemas for resource creation, XML and JSON representations

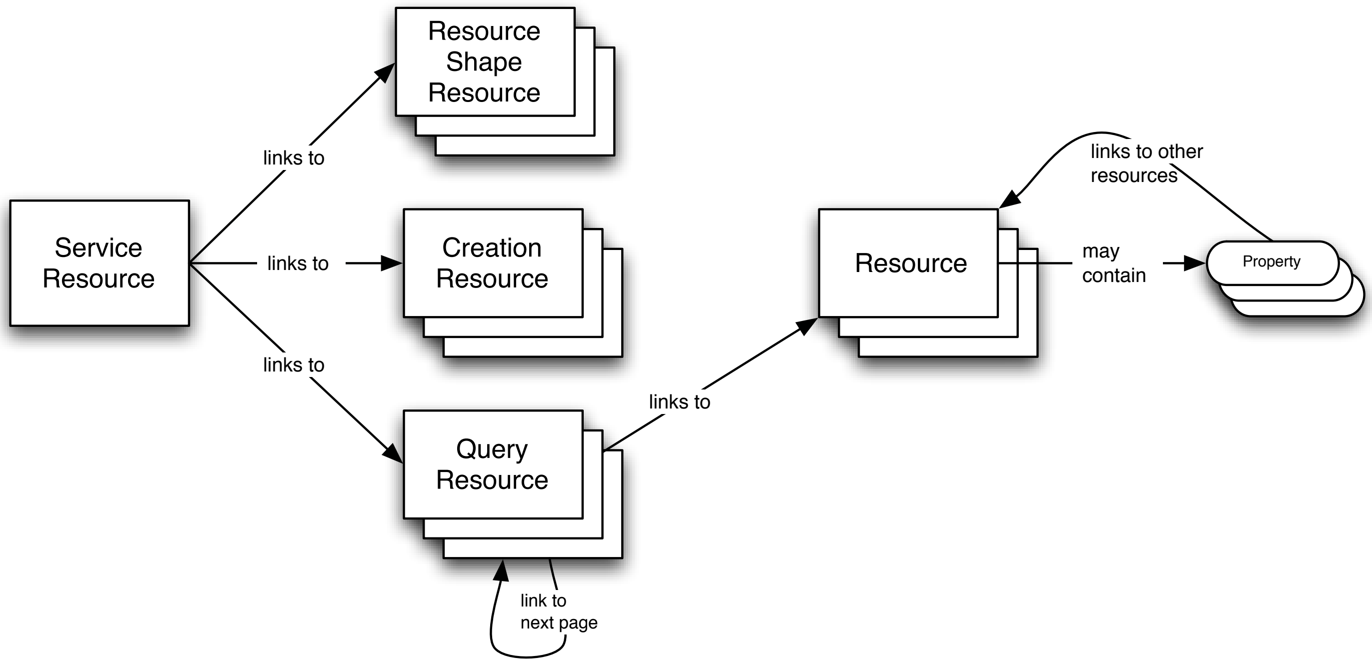
OSLC core spec approach

- Simple model
  - Everything is a resource with property values
- Rules for generating representations based on that simple model
  - RDF/XML required
  - Turtle, JSON, Atom allowed



OSLC core spec approach

- Also need three types of resources



Proposed core spec outline

- Overview
- OSLC Resources
- Resource Shape Resources
- Service Resources
- Query Resources
- Creation Resources
- Representations
- Authentication
- OSLC Common Patterns

OSLC Resource

- A resource that contains properties values meaningful to an OSLC Service.
- Normal rules of HTTP should apply:
  - Creation - POST (see also Creation Resource)
  - Retrieve - GET (see also Query Resource)
  - Update - PUT
  - Delete - DELETE

Resource Shape Resources

- A resource that describes a Resource Shape, listing the properties that are expected to be in resources of one specific shape.
- A set of Property definitions each with properties:
  - oslc:predicate (URI, Required) - predicate of property
  - oslc:datatype (URI, Required, Multi-valued) - datatype of property. May be String, Integer, Number, Boolean, or URI
  - oslc:minOccurs (integer, optional, default 0) - minimum number of instances allowed
  - oslc:maxOccurs (integer, optional, default is no limit) - maximum number of instances allowed
  - dc:title (String, optional) - title of property
  - dc:description (String, optional) - description of property
  - oslc:allowedValue (String, optional, multiple allowed) - value allowed for property
  - oslc:defaultValue (String, optional) - default value for property
  - oslc:maxSize (integer, optional, default is no limit) - maximum length of string property in characters
  - oslc:readOnly (boolean, optional, default is false) - true if property is read-only
- Spec provides set of common property definitions

Service Resources

- Resource that describes a set of OSLC Resources that together form an OSLC Service.
- Service can provide one or more:
  - Query Resources
  - Creation Resources
  - Resource Shapes

Creation Resources

- Resources can be created via normal HTTP POST
  - to a creation resource
- Response must include Location of created resource
- Response may include OSLC representation

Query Resources

- Conceptually a Query Resource is a family of resources all made available at the same base URI.
- Each resource in the family represents a set of resources that match a query criteria specified in the URI by the client.
- Using Query Syntax defined in CM 1.0 spec

OSLC Common Patterns

- Things that may not be suitable for cementing into a specification yet, but we’re ready to offer guidance
- Modeling Links
- Partial Update
- File & File Descriptor
- Hierarchical Web Content
- Delegated UI and Pickers
- Compact Rendering

OSLC Representations

- Because we have used a simple conceptual model of resources with properties, we can define simple rules for generating representations
- For the sake of inter-op OSLC Services MUST support RDF/XML representations of all resources
- They MAY support other representations and core spec will include rules for creating these:
  - Turtle
  - JSON
  - Atom

Next steps?

- Surface Core Spec draft/straw-man on Wiki
- Quickly merge in ongoing work on:
  - Query Syntax
  - Resource Shapes
- Review & Discuss Core Spec draft/straw-man
- Use mailing list and WG meetings for discussion
- Work towards convergence