Abstract. In this article, we discuss how geospatial data and services can be exposed over the Web. We introduce REST, outline a RESTful implementation of geospatial Web services that provides simple and open access to geospatial data over the Web using standard Web protocols. The term REST was introduced by Roy Fielding in his Ph.D. dissertation and describes an architecture style of networked systems. The motivation for REST has been to rely on the simplicity of the HTTP protocol and data exchange based on XML.

Keywords. RESTful, SOA, Web services

1. Introduction

A geospatial SOA is supported by the efforts of GIS (Geographic Information Systems) professionals who use GIS software [3], configure, and maintain GIS data, models, and applications. The authored content is published on a GIS server upon which services are delivered to support mapping, query, analysis, and more. Technically, Web services are modular applications that correspond to recognizable business functions and offer a set of protocols by which they can be published, discovered, and used in a standards-based way [1, 2]. Organizationally, Web services are simply information technology (IT) assets that are often used as the basis for integration strategies that fuse content and capabilities in support of various business processes and initiatives. Web services provide the building blocks upon which broader IT strategies are based, such as the implementation of a service oriented architecture (SOA). SOA and web services are two different things, but web services are the preferred standards-based way to realize SOA. This article provides an overview of geospatial SOA using the role of REST web services. Representational State Transfer (REST) is the next big trend in service-oriented architecture (SOA) development.

2. Deriving Web Services from SOA

Three Everyone knows roughly what a "web service" is, but there is no universally accepted definition. The definition of web service has always been under hot debate within the W3C Web Services Architecture Working Group. Despite the difficulty of defining web services, it is generally accepted that a web service is a SOA with at least the following additional constraints:

1. Interfaces must be based on Internet protocols such as HTTP, FTP, and SMTP.
2. Except for binary data attachment, messages must be in XML (EXtensible Markup Language) [6].

There are two main styles of Web services: SOAP web services and REST web services.

2.1 SOAP Web services

A SOAP (Simple Object Access Protocol) web service introduces the following constraints: Except for binary data attachment, messages must be carried by SOAP [6].

The description of a service must be in WSDL. A SOAP web service is the most common and marketed form of web service in the industry. Some people simply collapse "web service" into SOAP and WSDL services. SOAP provides "a message construct that can be exchanged over a variety of underlying protocols" according to the SOAP 1.2. In other words, SOAP acts like an envelope that carries its contents. One advantage of SOAP is that it allows rich message exchange patterns ranging from traditional request-and-response to broadcasting and sophisticated message correlations. There are two flavors of SOAP web services, SOAP RPC and document-centric SOAP web service. SOAP RPC web services are not SOA; document-centric SOAP web services are SOA.
2.2. SOAP RPC Web services

A SOAP RPC web service breaks the second constraint required by an SOA. A SOAP RPC Web service encodes RPC (remote procedure calls) in SOAP messages. In other words, SOAP RPC "tunnels" new application-specific RPC interfaces through an underlying generic interface. Effectively, it prescribes both system behaviors and application semantics. Because system behaviors are very difficult to prescribe in a distributed environment, applications created with SOAP RPC are not interoperable by nature. Faced with this difficulty, both WS-I basic profile and SOAP 1.2 have made the support of RPC optional. RPC also tends to be instructive rather than descriptive, which is against the spirit of SOA. Ironically, SOAP was originally designed just for RPC. It won't be long before someone claims that "SOAP" actually stands for "SOA Protocol".

2.3. REST Web services

The term REST was first introduced by Roy Fielding to describe the web architecture. A REST web service is an SOA based on the concept of "resource". A resource is anything that has a URI [6]. A resource may have zero or more representations. Usually, people say that a resource does not exist if no representation is available for that resource. A REST web service requires the following additional constraints:

Interfaces are limited to HTTP. The following semantics are defined [2]:

- HTTP GET is used for obtaining a representation of a resource. A consumer uses it to retrieve a representation from a URI. Services provided through this interface must not incur any obligation from consumers.
- HTTP DELETE is used for removing representations of a resource.
- HTTP POST is used for updating or creating the representations of a resource.
- HTTP PUT is used for creating representations of a resource.

The HTTP methods, PUT, GET, POST, and DELETE define a uniform interface for accessing resources (i.e. Create, Retrieve, Update, Delete or CRUD). HTTP is stateless request-response application protocol. Request and response messages are comprised of a command (method), a header, and a body. The interactions can be secured at the transport layer using the Secure Sockets Layer (SSL) protocol while the messages can be secured using encryption and a digital signature [1].

A resource representation is a self-descriptive, immutable physical snapshot of a resource. The most common resource representation is XML but other popular formats include JSON and XHTML. A representation may contain links to other resources (a.k.a. hypermedia). The use of hypermedia within representations allows consumers and providers to evolve independently over time.

A resource representation and its links correspond to a snapshot of the application state. Each interaction results in an updated application state; however, a resource’s state only changes in the case of a PUT, POST or DELETE. Furthermore, these interactions are said to be idempotent, meaning that duplicate consumer requests to create, update, or delete a resource are processed once and only once.

RESTful web services are secured by implementing authentication, authorization and auditing. Authentication is assurance of a consumer’s identity whereas authorization verifies that an identified consumer has the appropriate privileges. A privilege is comprised of a method and URI (e.g. POST /building/{building-id}).

Finally, auditing entails logging any action that updates the state of a given resource.

The first step in designing a RESTful web service is to answer several questions about your domain:

1. What are your entities (resources)?
2. How do your consumers want the data formatted (resource representations)?
3. Are there relationships between the resources (links)?
4. What identifies each entity (URI)?

In our facilities management web application, the resources include buildings and rooms. Most messages are in XML, confined by a schema written in a schema language such as XML Schema from W3C or RELAX NG. Simple messages can be encoded with URL encoding. Service and service providers must be resources while a consumer can be a resource.
REST web services require little infrastructure support apart from standard HTTP and XML processing technologies, which are now well supported by most programming languages and platforms.

3. Principles of Service Oriented Architecture

Services-Oriented Architecture (SOA) is a software architecture where functionality is grouped around business processes and packaged as interoperable services. Within well-defined interfaces and contracts among services, SOA connects different functional units (also called service) together of applications. The aim is a loose coupling of services with operating systems, programming languages and other technologies which underlie applications [3]. SOA is a sort of 3-layered architecture, mainly containing three logical parts: Service Consumer, SOA Infrastructure and Service Producer. Service is core of SOA. The consumer consumes service that produced by the service producer via SOA infrastructure.

4. Comparing REST and SOAP "styles"

More and more people are getting turned on to the advantages of using REST as a higher-level abstraction for networked applications, often comparing it favorably to SOAP and Web services. REST and SOAP are different technologies for different usage scenarios.

1. REST is for simple request/response needs.
2. SOAP is for more complex request/response needs.

SOAP is Service Oriented Architecture Protocol, an XML-based messaging protocol useful for performing RPC-style (Remote Procedure Call) request-response dialogues. REST is architecture style, it is an architectural model like the web itself is. REST is a style of creating applications which works on same principles as the web [3].

5. Why REST?

REST is easy to implement and has many highly desirable architectural properties: scalability, performance, security, reliability, and extensibility. Those characteristics fit nicely with the modern business environment, which commands technical solutions just as adaptive and agile as the business itself.

Two more recent approaches are: i) the W3C's Simple Object Access Protocol (SOAP) which is supported by application development tool makers such as IBM, BEA Systems, and Microsoft; and ii) REST which has been used by Amazon, Google, and others to create interfaces to their Web services [7]. For example, Amazon's web services have both SOAP and REST interfaces, and 85% of the usage is on the REST interface. One of the most basic examples of a REST-based SOA is Amazon's affiliate network (or for that matter any e-commerce affiliate network). Amazon's affiliate partners simply paste a small snippet of code into their website and then become part of a distributed web service in which they display Amazon's goods and Amazon then pays commissions on any sales that occur as the result of their referrals. This service, now with tens of thousands of participants, is all done by leveraging the existing web infrastructure. Amazon does offer more complex SOAP-based
web services, but these services are have been adopted by only a small set of their affiliates due to their complexity.

Another great example of a REST-based SOA is Google’s AdSense network [7].

6. How REST-ful are OGC Services?

The term REST stands for Representation State Transfer and was coined in the PhD thesis of Fielding [2]. REST in my view proposes principles about web-based software architecture design (a.k.a. architectural styles). It does not give guidelines, but extracts certain properties (called constraints), which should be inherited from these REST-ful components (such as Web Services). These properties reflect also the basic concepts of the Web. So a REST-full web service has to be:

- stateless
- talking over HTTP (POST and/or GET)
- Using URLs

Even these properties are somehow applied on a web service technology view. Most of the articles on the Web somehow do forget about that. So for instance, assuming that REST is XML based or is on the same stage as SOAP, is not the case. However it can be compared to SOAP/WSDL, if you want to compare them on a level of fine-grain web architecture design. Doing so, it becomes clear that, REST seems to be sufficient in most of the projects and it only has to be extended, if real enterprise applications have to be bringing to the market (and then use SOAP/WSDL). Ok, so far, that is theory. Now back to the question. If I only look at the specifications of OGC Services I would definitely say that OGC Services (as WMS, WFS, WCS) are actually fulfilling these REST-properties. But you barely find any evidence, that OGC Services are REST-full. A recent post at the Surveying, Mapping and GIS blog links somehow REST and OGC, but does not identify a clear link, between the both. Regarding the second question, I would say, that the question is relevant, to assess the development stage of such a specification. You can clearly see, that OGC pushes their specifications towards SOAP/WSDL if they are mature enough (e.g. WMS). Additionally it shows, that OGC’s specification process (at least the results) are quite software driven and always attached to a real implementation scenario.

7. Example Realizing SOA with REST Web Services

Web When implementing Geospatial SOA we need to support functionalities for:

- Access data and geometry attributes
- Access image of data
- Edit of spatial and non-spatial attributes
- Enable spatial queries on data
- Enable different representation of data: image, html, kml, xml, GeoJson

Those functionality are implemented inside Restful Geospatial Web Service. To access data or execute geospatial service request is send to Restful web service. Request are executed using standard HTTP methods and parameters for execution are send as URL query parameters. HTTP GET method will return representation of data in representation format defined as part of URL. Spatial and Data filter can be set using URL query parameter ‘filter’. Commonly used spatial filter is ‘BBOX’ which allows to return data included inside particular rectangular window.

Examples of URI requests and responses are:

HTTP GET
http://www.sl-king.si/mgrest/rest/data/parcel/29.html

This request will return HTML representation of parcel with id 29.

If we want to get XML representation of parcel we would execute

HTTP GET
http://www.sl-king.si/mgrest/rest/data/parcel/29.xml

In similar way we can get image representation of parcel by requesting png as representational format to return.

HTTP GET
http://www.sl-king.si/mgrest/rest/data/parcel/29.png
To filter data by some parameter we can use:

HTTP GET
http://www.sl-
ing.si/mgrest/rest/data/parcel/29.html?filter=NAME='SomeName'

To filter data by spatial parameter we can use:
http://www.sl-
ing.si/mgrest/rest/data/parcel/29.html?BBOX=1.1,1.1,2.2,2.2

Where BBOX represents rectangular window (1.1,1.1) (2.2,2.2)

What is very powerful about Web Services is that we can combine more services or servers to execute one task. For example we can have data of buildings served by one service and we can have parcels served by another server. We can create request which will return parcels which are under particular building.

URI would look like:

http://www.sl-
ing.si/mgrest/rest/data/parcel/32.html

This request will return parcels under building with ID 32. While executing this request one service will call another service to get building with that id. We could also combine other filters and parameters while executing this requests. If we would like to update data we need to use PUT, POST or DELETE HTTP methods. This methods can be called from different clients and environments. Today probably very interesting is using Restful Geospatial Web Services in AJAX application. For that purpose we can use GeoJSON to query or update data.

To get data URL would look like:

HTTP GET
http://www.sl-
ing.si/mgrest/rest/data/parcel/29.json

and GeoJSON response is:

```json
{
  "type": "FeatureCollection",
  "members": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Polygon",
        "coordinates": [[[87.708336985426598, 43.747379333687], 
                         [87.70788515931596, 43.7475654407402],
                         [87.706685671813503, 43.7471052779497],
                         [87.707149749538601, 43.746849468228802],
                         [87.7071470443029, 43.74683705831199],
                         [87.70714493725794, 43.746825487454998],
                         [87.707143450413795, 43.74682070477297],
                         [87.707142566220099, 43.74679523135103],
                         [87.707142292241699, 43.746786961001],
                         [87.707142628818204, 43.74774396664499],
                         [87.707143575531603, 43.74671854722202],
                         [87.707145131206502, 43.74679434738102],
                         [87.707147293911504, 43.74676876244803],
                         [87.707150060962206, 43.74724473713698],
                         [87.707153428923803, 43.7471249538197],
                         [87.70715793615901, 43.74669991904397],
                         [87.707161950171197, 43.7468779323202],
                         [87.707167092772295, 43.74667599506399],
                         [87.707172815197893, 43.7466390457999],
                         [87.707179110291307, 43.7466234895599],
                         [87.70718597023915, 43.746604697437701],
                         [87.707193386527095, 43.746692431339],
                         [87.707201349949898, 43.74661818347699],
                         [87.707209850623599, 43.74660724383699],
                         [87.707218877997406, 43.74596510195502],
                         [87.707228402868689, 43.74586022782799],
                         [87.707238467387796, 43.7456758754551],
                         [87.707249000509053, 43.74565811080302],
                         [87.7072600208693, 43.74556111938399],
                         [87.707272510132794, 43.74564670023081],
                         [87.707528978270503, 43.745329607349],
                         [87.708761752353098, 43.747161709664397],
                         [87.708651432474298, 43.742900469497],
                         [87.708637362745804, 43.7425976232097],
                         [87.708622790777994, 43.74227060785298],
                         [87.708607733831798, 43.74280185956998],
                         [87.708592209743003, 43.74289836785399],
                         [87.708576236902299, 43.74290901015996],
                         [87.708559834224403, 43.74307941295503],
                         [87.708543021145005, 43.74316377140201],
                         [87.708525817578106, 43.74323488557502],
                         [87.708508243902003, 43.74331966057303],
                         [87.708490320933706, 43.74339100663702],
                         [87.708472069904005, 43.74345783925198],
                         [87.7084535124322, 43.74352007925102],
                         [87.708436708050943, 43.74357765290801],
                         [87.708415566428499, 43.743630492202],
                         [87.708396222845906, 43.74376853403938],
                         [87.70836662666396, 43.74372712194297],
                         [87.7083569090602, 43.74376000468101]]
```
To update data we would need to execute HTTP POST method and send GeoJSON in body of request

HTTP PUT
http://www.sking.si/mgrest/rest/data/parcel/29.json
with updated GeoJSON included in body of request.

RESIDENTIAL ZONE","zone_value":"RS-1"});}}

8. Conclusion

The Both the REST and SOAP styles for the delivery of Web services are well established in developer consciousness. SOAP has a much more elaborate complex of standardization efforts and open source toolkits. Furthermore many IDEs now provide for automatic generation of SOAP based interfaces for existing code. SOAP is also indicated if you need to publish your Web service with WSDL (Web Service Definition Language) or you need security functions such as message signing and encryption. On the other hand, if you need to make some information public with a simplified interface and without using a lot of processor power, REST may be for you.

9. References