

Restafarian: 80% of developers use the XYZ REST API because they find it easier to deal with.

Tool Vendor: That's not really REST. It only uses GET and POST and breaks rule #5034

Restafarian: Oh yea, well SOAP/WS-* sucks.

Tool Vendor: Go write some PHP you dirty hippie!



Understanding REST-Based Services: Simple, Scalable, and Platform Independent

**DevCares
April 24th, 2009**

Table of Contents

- ▶ **Background and Controversy**
 - Background & Business Problems
 - REST vs. SOAP Debate

- ▶ **Introduction to REST**
 - REST as an Architectural Style
 - Trend in the Marketplace

- ▶ **REST Architectural Style and Principles**
 - The REST Triangle
 - Benefits and Criticisms
 - Comparing REST to SOAP & WS-*
 - Demo

- ▶ **Building RESTful services with ADO.NET Data Services**
 - Introducing ADO.NET Data Services
 - Deeper Look at ADO.NET Data Services
 - Demo

- ▶ **Recommendation for Using RESTful Services**

- ▶ **Appendix**
 - External Links

A major goal of web services is to provide a way to create and interact with systems across a network

Background

- ▶ According to IBM, web services are “self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains”
- ▶ The web services model that most people are familiar with consists of using XML messages that follow the SOAP standard in methods that are written in WSDL, which provide a machine-readable description of available service operations
- ▶ Many organizations exist to promote and create specifications and standards related to web services

Business Problems

- ▶ Businesses must consider many things when deciding how to integrate systems and expose data to the rest of their supply chain
- ▶ A balance must be achieved between competing requirements like reliability, security, scalability, and performance for the integration to be successful and useful

Recently, the concept of RESTful services have emerged to challenge SOAP as the go-to solution for creating interoperable services

- ▶ Restafarians, hardcore proponents of REST, denounce SOAP and the WS-* stack with religious fervor
 - SOAP adds unnecessary complexity
 - SOAP interfaces are brittle and fragile
 - SOAP abuses HTTP
 - SOAP is slow
 - SOAP is not based on open source implementations
- ▶ Supporters of SOAP/WS-* attack Restafarians by pointing out flaws in the implementations of REST
 - REST is just an abstraction, no real-world RESTful system exists today
 - REST is hard to understand and implement correctly
 - REST services cannot handle the need for complex business logic
 - REST is not simple

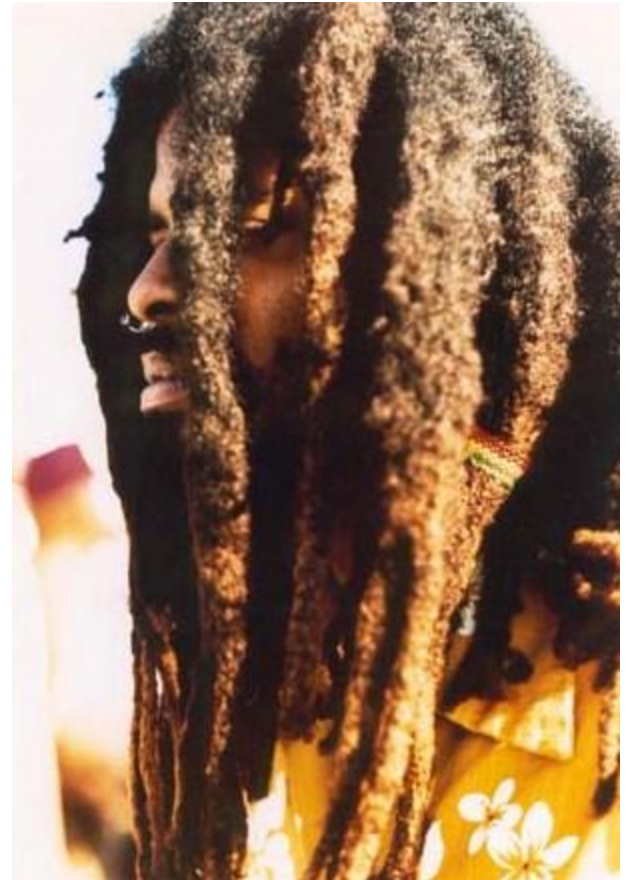


Table of Contents

▶ **Background and Controversy**

- **Background & Business Problems**
- **REST vs. SOAP Debate**

▶ **Introduction to REST**

- **REST as an Architectural Style**
- **Trend in the Marketplace**

▶ **REST Architectural Style and Principles**

- **The REST Triangle**
- **Benefits and Criticisms**
- **Comparing REST to SOAP & WS-***
- **Demo**

▶ **Building RESTful services with ADO.NET Data Services**

- **Introducing ADO.NET Data Services**
- **Deeper Look at ADO.NET Data Services**
- **Demo**

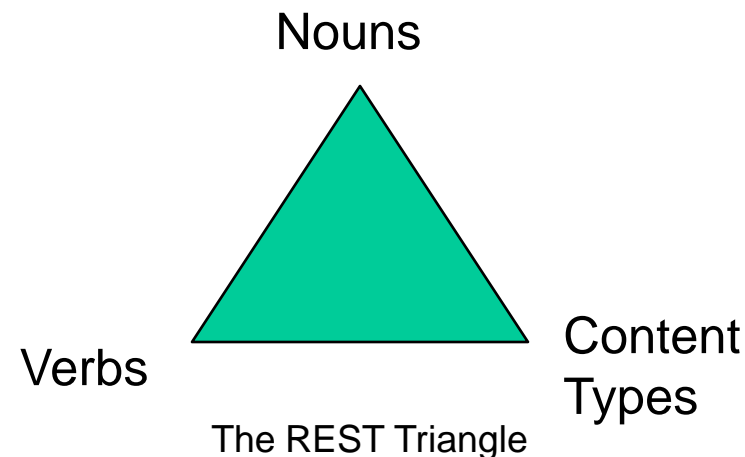
▶ **Recommendation for Using RESTful Services**

▶ **Appendix**

- **External Links**

Representational State Transfer (REST) is an architectural style that defines principles that enable the creation of scalable distributed software systems

- ▶ REST initially came on the market as an alternative to developing services using the SOAP/WS-* specifications
 - SOAP based services are not integrated with each other and are not designed so that third parties can easily integrate with them
 - Two services that implement the same WS-* specification may not be able to communicate with each other
- ▶ According to Roy T. Fielding in his doctoral dissertation, REST focuses on key components that enable easy integration and interaction with other systems
 - Scalability of the interaction between systems
 - Generalization of interfaces
 - Independent deployment of components
 - Communication between intermediary components that provide additional functionality
- ▶ To achieve its goals REST leverages principles that the modern internet employs (HTTP, URIs)
 - Identify all resources through a URI
 - Use links to refer to resources
 - Use standard methods
 - Resources can have multiple representations
 - Communicate without maintaining state
- ▶ REST approaches the need for interoperability and communication between systems through separation of concerns
 - Nouns – identifies resources (URI)
 - Verbs – standard HTTP methods (GET, POST, PUT, DELETE)
 - Content Types – types of content (MIME types)



Large consumer facing companies are investing in REST, and we need to watch the trends so that we understand when the technology would provide value to our customers

- ▶ Yahoo
 - “We believe REST has a lower barrier to entry, is easier to use than SOAP, and is entirely sufficient for these services.”
- ▶ Facebook
 - “Nearly any computer language can be used to communicate over HTTP with the REST server.”
- ▶ Flickr
 - “The Flickr API supports many protocols including REST, SOAP, XML-RPC.”
- ▶ Amazon
 - “Uses standards-based REST and SOAP interfaces designed to work with any Internet-development toolkit.”
- ▶ Google
 - “As of December 5, 2006, we are no longer issuing new API keys for the SOAP Search API”
- ▶ ebay
 - “The Shopping API is optimized for response size, speed and usability.”
- ▶ digg
 - “The API returns Digg data in a form that can be easily integrated into an application or a web site”



Table of Contents

▶ **Background and Controversy**

- **Background & Business Problems**
- **REST vs. SOAP Debate**

▶ **Introduction to REST**

- **REST as an Architectural Style**
- **Trend in the Marketplace**

▶ **REST Architectural Style and Principles**

- **The REST Triangle**
- **Benefits and Criticisms**
- **Comparing REST to SOAP & WS-***
- **Demo**

▶ **Building RESTful services with ADO.NET Data Services**

- **Introducing ADO.NET Data Services**
- **Deeper Look at ADO.NET Data Services**
- **Demo**

▶ **Recommendation for Using RESTful Services**

▶ **Appendix**

- **External Links**

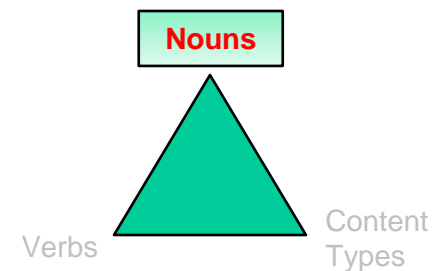
Consumers are interested in the data in the system and REST based services are centered around nouns

Overview

- ▶ Nouns (the data) represent the resources or objects that can be changed or retrieved and are the cornerstone of the REST architectural style
- ▶ Resources are the logical objects that are in the system
 - Ex: For an online book retailer, books, publishers, and reviews would all be considered nouns
- ▶ Resources can have multiple representations
 - Ex: The online book retailer could return the book in either a JSON format that is easily parsed by an AJAX based client, or just POX that can be parsed by any client

Guiding Principles

- ▶ REST proposes making nouns as small as possible to decrease message size and increase performance
- ▶ Retrieving a resource should not in and of itself change the resource, instead pass around representations of the resource
- ▶ Nouns should be as self-descriptive as possible so that clients can drill down and interpret the data without relying on requests made before or after it
 - Ex: Each book at our online book retailer would have links to other books, reviews, publishers, etc. that allow the user to browse to other related information



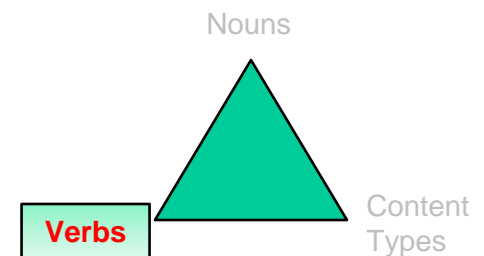
REST advocates using a minimum set of actions against nouns to reduce service complexity and to keep the barriers to adoption low

Overview

- ▶ REST describes application behavior using the basic HTTP methods, which are similar to the standard CRUD operations:
 - GET (c**R**ud)
 - Can be used to retrieve the entire object, or just a part of the object
 - Ex: Retrieve *Ender's Game* by Orson Scott Card, or the first 5 books from the New York Times Best-Seller list
 - Can be repeated multiple times without changing the resource, which allows for the indexing and/or caching of the results
 - PUT (cr**U**d)
 - Is a request to replace a resource with a new resource
 - Ex: Update the 3rd review about Guns, Germs, and Steel with a revised review from the critic
 - No partial updates are allowed
 - POST (**C**rud)
 - Post can be used to perform a variety of different functions
 - Ex: Insert new reader comment about *User Stories Applied* by Mike Cohn
 - DELETE (cru**D**)
 - Is a request to delete a resource completely
 - Ex: Remove the negative comment about *RESTful Web Services* by Richardson and Ruby
 - No partial deletes are allowed

Guiding Principles

- ▶ Using a standard set of actions (Verbs) allows ANY client that implements the protocol to communicate with your system
 - Ex: a third party distributor of books that wrote a system that implements and understands HTTP requests would be able to easily integrate with our online book retailer
- ▶ Only use universal verbs that can be applied to all nouns



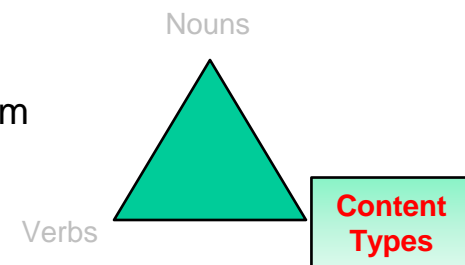
REST emphasizes standard content types between systems to achieve maximum interoperability

Overview

- ▶ The content type describes the potential representations of an object or resource (XML, JSON, ATOM)
- ▶ When standard content types and verbs are used, any client can interact with any RESTful service in the world
- ▶ Allowing multiple content types for a single resource lets the client request the content types that make the most sense for the situation
- ▶ Standardization of content types allows other services to deal with the data without necessarily understanding what the data actually means or where it comes from

Guidelines

- ▶ Use standard content types when possible to ensure maximum interoperability
- ▶ When creating custom content types, follow three rules of thumb:
 - Produce as few as possible
 - Make the content type as widely understood as possible
 - Extend existing formats if possible
- ▶ Design the service so that the addition of new content types does not affect the system architecture as a whole
 - Ex: adding a PDF content type to a book in our online book retailer should not change how verbs interact with the book



Demo: Interacting with a RESTful service

```
http://twitter.com/statuses/user_timeline.xml?screen_name=devcaresdemo - Windows Internet Explorer
http://twitter.com/statuses/user_timeline.xml?screen_name=devcaresdemo
Twitter / Home http://twitter.com/stat... X
<?xml version="1.0" encoding="UTF-8" ?>
- <statuses type="array">
- <status>
  <created_at>Fri Apr 24 02:54:26 +0000 2009</created_at>
  <id>1600374287</id>
  <text>moving on to ado.net data services demo</text>
  <source>web</source>
  <truncated>>false</truncated>
  <in_reply_to_status_id />
  <in_reply_to_user_id />
  <favorited>>false</favorited>
  <in_reply_to_screen_name />
- <user>
  <id>16861442</id>
  <name>Charles Knight</name>
  <screen_name>DevCaresDemo</screen_name>
  <location />
  <description />
  <profile_image_url>http://s3.amazonaws.com/twitter_production/profile_images/62425764/224-036_Bob-Marley-Posters_normal.jpg</profile_image_url>
  <url />
  <protected>>false</protected>
  <followers_count>5</followers_count>
  <profile_background_color>EDECE9</profile_background_color>
  <profile_text_color>634047</profile_text_color>
  <profile_link_color>088253</profile_link_color>
  <profile_sidebar_fill_color>E3E2DE</profile_sidebar_fill_color>
  <profile_sidebar_border_color>D3D2CF</profile_sidebar_border_color>
  <friends_count>2</friends_count>
  <created_at>Mon Oct 20 03:42:44 +0000 2008</created_at>
  <favourites_count>0</favourites_count>
  <utc_offset>-21600</utc_offset>
  <time_zone>Central Time (US & Canada)</time_zone>
  <profile_background_image_url>http://s3.amazonaws.com/twitter_production/profile_background_images/3245127/CVF031-BOB-MARLEY-colours.jpg</profile_background_image_url>
  <profile_background_tile>>true</profile_background_tile>
  <statuses_count>6</statuses_count>
  <notifications>>false</notifications>
  <following>>false</following>
</user>
</status>
```

There are pros and cons to developing services with a RESTful approach when compared to a SOAP/WS-* based approach

Criteria	Considerations	REST	POX / HTTP	SOAP
Functionality	Application's ability to deliver its required capabilities	2	1	4
Usability	User's productivity when working with the application	4	2	2
Affordability	Application's overall cost including acquisition and on-going maintenance	4	4	1
Maintainability	Level of effort required to keep application running while in production including problem resolution and on going support	2	2	2
Flexibility	Ability to accommodate additional business processes or changes in functionality	4	2	4
Scalability	Ability of application to support additional users while meeting quality of service goals	4	2	2
Interoperability	Ability of the system to interact effectively with other systems	4	4	2
Security	Ability of the application to prevent unauthorized disclosure, loss, modification or use of its data or functionality	1	1	4
Market Support	Application's conformance with existing and emerging infrastructure with internal and external standards	3	1	4
Overall		3	1	3

Table of Contents

- ▶ **Background and Controversy**
 - Background & Business Problems
 - REST vs. SOAP Debate

- ▶ **Introduction to REST**
 - REST as an Architectural Style
 - Trend in the Marketplace

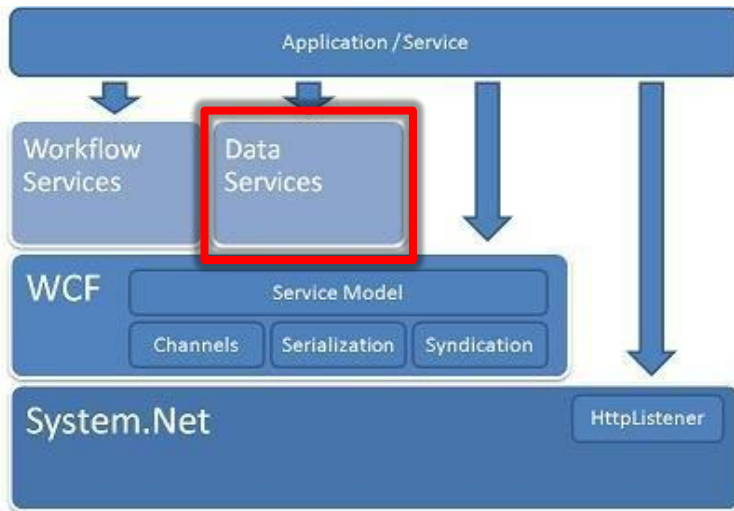
- ▶ **REST Architectural Style and Principles**
 - The REST Triangle
 - Benefits and Criticisms
 - Comparing REST to SOAP & WS-*
 - Demo

- ▶ **Building RESTful services with ADO.NET Data Services**
 - Introducing ADO.NET Data Services
 - Deeper Look at ADO.NET Data Services
 - Demo

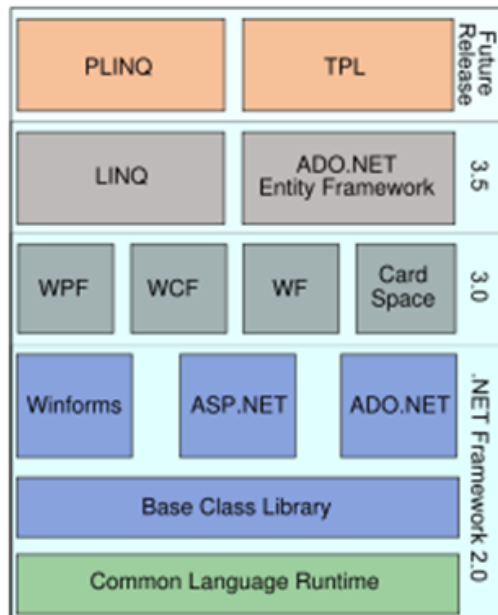
- ▶ **Recommendation for Using RESTful Services**

- ▶ **Appendix**
 - External Links

ADO.NET Data Services is a Microsoft framework that allows developers to create RESTful services



.NET Services Platform

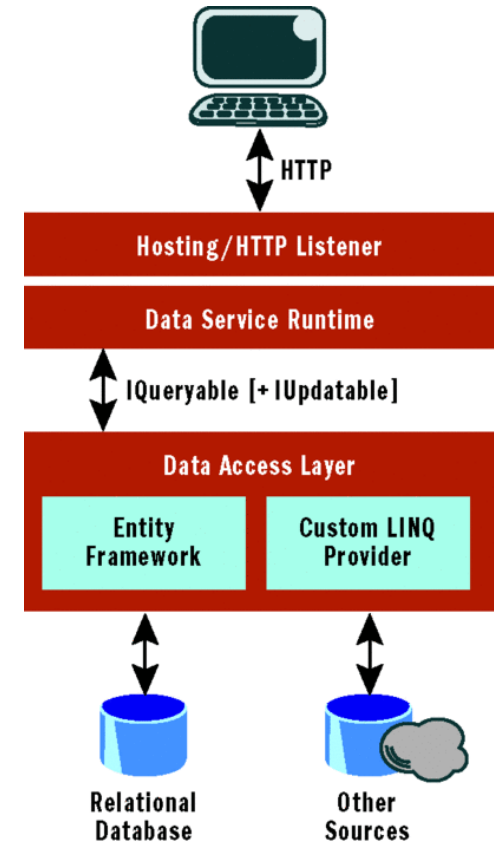


The .NET Framework Stack

- ▶ As part of the .NET 3.5 Framework, Microsoft built support for REST into the WCF Web Programming Model
- ▶ The functionality included in the WCF Web Programming Model centers around many of the principles of REST
 - Support for URIs
 - Standard HTTP verbs
 - Formats and ContentTypes
- ▶ The ADO.NET Data Services framework provides patterns and libraries that enable the creation and consumption of data driven services for the web
 - Uses an Entity Data Model as a conceptual model to describe resources and associations
 - Enables a simple addressing scheme for entities and properties
 - Supports entity representation in JSON and AtomPub formats
 - Shows clear separation of service and data store

ADO.NET Data Services allows developers to build applications that make it easy to find, manipulate, and deliver data through the use of simple URIs

- ▶ ADO.NET Data Services can be built on top of any data source
 - Support for Entity Framework Models
 - Support for other data sources that implement `IQueryable<T>` and `IUpdatable`
- ▶ Query string parameters used with URIs help to control the output of data returned from the service
- ▶ Service operations can be used to add additional business logic to the data service
 - Methods are exposed and identified by a URI
 - Method parameters are passed in through the query string
 - Methods are tagged to enable invocation using GET, PUT, POST, or DELETE requests
- ▶ Query interceptors can alter data, enforce authorization requirements, or terminate operations in both requests and responses
 - Interceptors cannot accept parameters
 - Interceptors use lambda expressions
- ▶ Offline-Enabled Data Services (Astoria Offline)
 - Can be used to implement 2-way synchronization of data
 - Removes client reliance on service availability
 - Currently in Alpha preview



Demo: Build and consume an ADO.NET data service

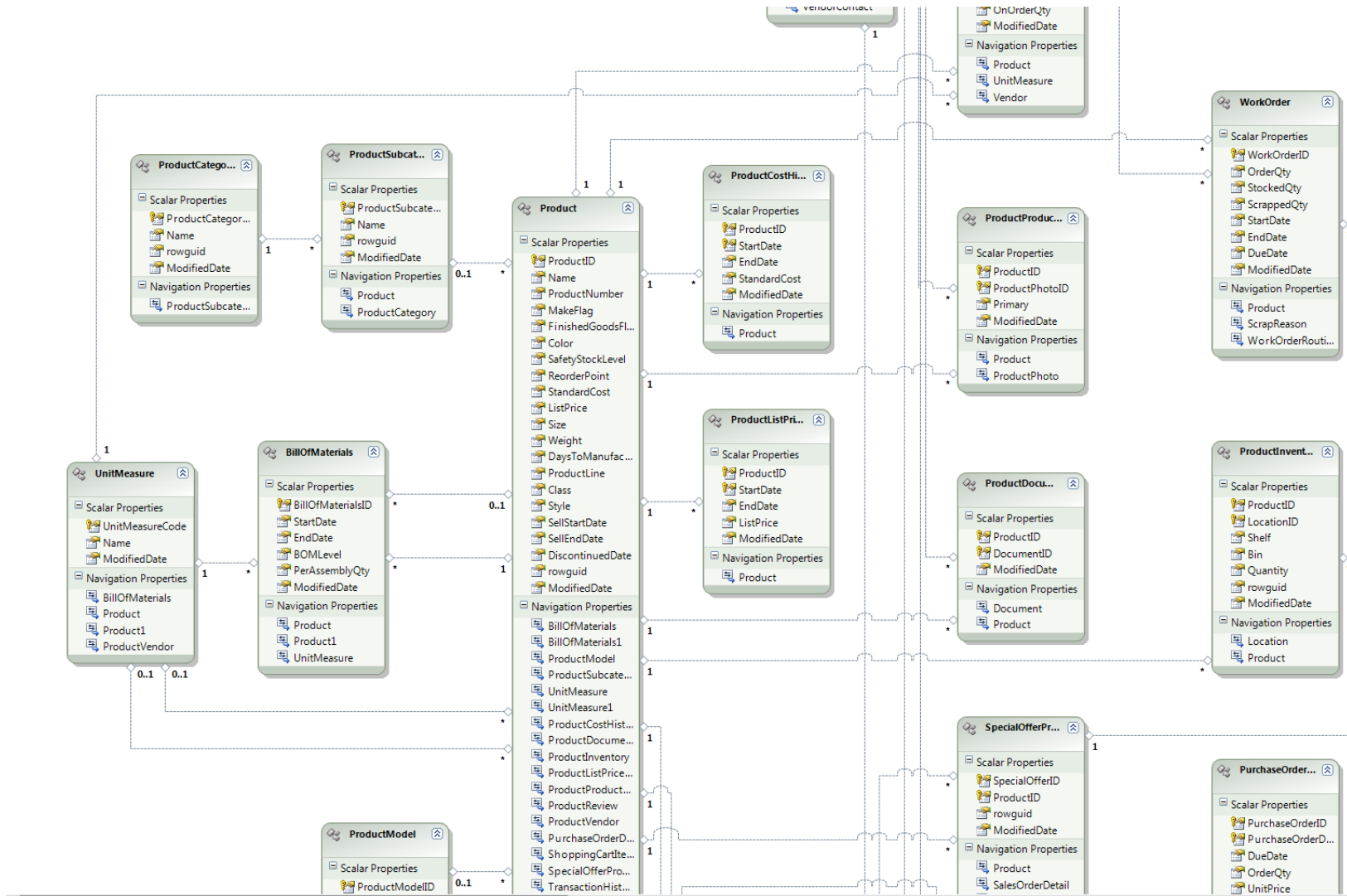


Table of Contents

- ▶ **Background and Controversy**
 - Background & Business Problems
 - REST vs. SOAP Debate

- ▶ **Introduction to REST**
 - REST as an Architectural Style
 - Trend in the Marketplace

- ▶ **REST Architectural Style and Principles**
 - The REST Triangle
 - Benefits and Criticisms
 - Comparing REST to SOAP & WS-*
 - Demo

- ▶ **Building RESTful services with ADO.NET Data Services**
 - Introducing ADO.NET Data Services
 - Deeper Look at ADO.NET Data Services
 - Demo

- ▶ **Recommendation for Using RESTful Services**

- ▶ **Appendix**
 - External Links

Services based on REST and the SOAP / WS-* stack are not mutually exclusive

- ▶ A service based on the REST architectural style is a good design decision when:
 - There is a great need for your service to be **easily integrated** with third party systems
 - The solution caters to a **large user base**, with each user having different and evolving needs
 - Potential exists for **unknown clients** to interact with the service without the designers knowing about it
 - The service needs to be **massively scalable** and distributed
 - The data model lends itself to being abstracted into resources
 - The operations on the resources center around the **basic CRUD** operations
- ▶ Typically these scenarios exist for those applications that are built and exposed to the general public on the Web for the purpose of allowing customers and partners to interact with internal systems
- ▶ A service built using the SOAP and WS-* stack would be preferable when:
 - The problem takes place in a **“closed world”** where you know all of the users of the system and can communicate directly to them changes to the underlying model and structure of the interface
 - Integrating systems with the service needs to follow a **controlled process**
 - The underlying data model is **complex**
 - The **security architecture** of the entire system is complicated, or security requirements are high
 - There is need for **transaction oriented processing** to occur, with the potential for rollbacks
 - The service is addressing a business workflow that has **many interrelated steps**
- ▶ We typically see these scenarios where we are creating custom intranet applications that integrate with or replace legacy systems

“Why must we make everything into a rivalry? Why can't we just enjoy the best of all worlds?” – Jon Udell

Here are some common misconceptions that I found while researching the SOAP vs REST debate

- ▶ REST is an application protocol
- ▶ REST is a standard
- ▶ Anything that uses XML over HTTP is RESTful
- ▶ RESTful services can only use HTTP as a transport mechanism
- ▶ Using URIs to identify objects makes a service RESTful
- ▶ REST and SOAP cannot coexist
- ▶ You have to adhere to the REST style at all times to gain the benefits of REST
- ▶ REST cannot support complex transactional business workflows
- ▶ REST cannot support complex security requirements



Questions?



Appendix: External links and references

- ▶ For those wanting to read about REST from the father himself:
 - <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
- ▶ The definitive resource for building real world RESTful services:
 - <http://www.amazon.com/Restful-Web-Services-Leonard-Richardson/dp/0596529260/>
- ▶ Humorous (and probably fictional) but quite informative dialogue describing REST:
 - <http://tomayko.com/writings/rest-to-my-wife>
- ▶ Links to help get started building RESTful services
 - <http://www.codeplex.com/aspnet/Wiki/View.aspx?title=WCF%20REST&referringTitle=Home>
 - <http://www.microsoft.com/downloads/details.aspx?FamilyID=3e3d4eaf-227b-4ad3-ad0d-3613db8aa9df&displaylang=en>
- ▶ Brian's Blog
 - <http://borrell.parivedasolutions.com>