

RESTful web services Sriram Narayan

Introduction

The world of web services has been on a fast track to supernova ever since the architect astronauts spotted another meme to rocket out of pragmatism and into the universe of enterprises.

But, thankfully, all is not lost.

A renaissance of HTTP appreciation is building and, under the banner of REST, shows a credible alternative to what the merchants of complexity are trying to ram down everyone's throats; a simple set of principles that every day developers can use to connect applications in a style native to the Web.

- DHH (Creator of Ruby on Rails)

Business reasons to use REST

- Web is moving towards easier content syndication/data access and away from "website lock in".
 - newspapers allow RSS/Atom feeds
 - Amazon/EBay APIs, Google Maps and YouTube.
 - <http://www.programmableweb.com/apis>
 - Interconnected beyond hyperlinks e.g. analytics, ads, Mashups.
- There was a time when deep linking incited lawsuits. Today deep linking helps your PageRank!
- Ease of adoption becomes important. REST is easy to consume.
- Invites aggregators and mashups to use the service - drives business for retail apps
- Arguably lower cost of implementation

Applicability

- Sweet spots
 - Public facing websites/services: news, online retailing, community services.
 - Within the enterprise
- Weaker case for:
 - Personal services like email, internet banking.
 - Specialized business applications like InsureCom

Context setting : what is a service?

- Multiple consumers with varying lifecycles, control domains
- Autonomous evolution, see <http://www.google.com/apis/adwords/developer/AccountService.html>
- Just having a presentation layer make a XML/SOAP over HTTP call to the server side doesn't turn the server side into a service.

A simple flight booking service

3 pseudo WSDLs...

```
http://airdeccan.net/reservations/bookingService.asmx  
Flight[] getFlights(Date journeyDate, Place from, Place to)  
BookingConfirmation bookFlight(String flightNumber, int numberOfSeats)  
BookingConfirmation rescheduleFlight(String bookingReference, Date rescheduledDate)  
BookingConfirmation cancelBooking(String bookingReference)
```

```
http://goair.com/flightService  
FlightAvailabiltyResponse lookupFlights(FlightAvaiabilityRequest enquiry)  
ReservationResponse makeReservation(ReservationRequest request)  
ReservationResponse changeReservation(RescheduleRequest request)  
ReservationResponse cancelReservation(CancellationRequest request)
```

```
http://indigo.com/websvcgateway  
Response process(Request request)
```

and their RESTful counterparts

```
http://airdeccan.net/reservations/bookingService.asmx  
Flight[] getFlights(Date journeyDate, Place from, Place to)  
BookingConfirmation bookFlight(String flightNumber, int numberOfSeats)  
BookingConfirmation rescheduleFlight(String bookingReference, Date rescheduledDate)  
BookingConfirmation cancelBooking(String bookingReference)
```

```
GET http://airdeccan.net/reservations/v1/flights?journeyDate={journeyDate}&from={origin}&  
to={destination}  
POST http://airdeccan.net/reservations/v1/booking  
PUT http://airdeccan.net/reservations/v1/booking/{bookingReference}  
DELETE http://airdeccan.net/reservations/v1/booking/{bookingReference}
```

```
http://goair.com/flightService  
FlightAvailabiltyResponse lookupFlights(FlightAvaiabilityRequest enquiry)  
ReservationResponse makeReservation(ReservationRequest request)  
ReservationResponse changeReservation(RescheduleRequest request)
```

ReservationResponse cancelReservation(CancellationRequest request)
GET http://goair.com/flightservice/v1/flights?travelDate= {travelDate}& from = {origin}& to= {destination} POST http://goair.com/flightservice/v1/reservation PUT http://goair.com/flightservice/v1/reservation/{PNR} DELETE http://goair.com/flightservice/v1/reservation/{PNR}
http://indigo.com/websvcgateway Response process(Request request)
GET http://indigo.com/websvcgateway/v1/oneWayJourneys?dateOfTravel = {travelDate}& from = {origin} & to = {destination} POST http://indigo.com/websvcgateway/v1/{flightNumber}/reservation PUT http://indigo.com/websvcgateway/v1/{flightNumber}/reservation/{PNR} DELETE http://indigo.com/websvcgateway/v1/{flightNumber}/reservation/{PNR}

- Service interface = Semantics + Signature
- Signature = Endpoint(s) + Data Types + Actions (i.e. operations or verbs)

REST essentials

- Uniform interface (4 actions)
- Verbs become nouns
- Every piece of information has its own URL (from REST Easy)
- Content Type signifies Data Type
- Endpoints are potentially infinite
- Do the demo here?

HTTP Primer

- Important Request Headers
 - Accept (JSON, XHTML), Authorization, If-Modified-Since
- Important Response Headers
 - Content-Type, WWW-Authenticate, Last-Modified, Location
- Classes of Response codes
 - Success - 200 to 207
 - Redirect - 300 to 307
 - Client side Error - 400 to 417 **client error codes**
 - Server side Error - 500 to 505

Technical aspects of REST

- Addressability
 - Hyperlinks in the response!
 - Statelessness
 - Caching
 - Security
-
- This is CRUD #%\$@! What about object orientation?
 - Server side could be declarative (event-driven) or imperative. e.g. canceling a booking raises an event that is listened to by flight (for recovering seats) and by Confirmations Gateway (to send email to user)
-
- Consumption modes
 - Directly consumed by browser with CSS and a dash of JS
 - Consumed by AJAXy browser
 - Browser/Thick client talks to presentation-server that talks to RESTful service

REST + Microformats = Semantic Web!

- already possible - adding feeds to your reader, mailto, callto
- The possibilities ...
 - adding events to calendar
 - contact info into address books
 - order a specified book at a local bookstore or request them at librarywala.com
 - add a blogged about movie to your rental queue on 70mm
 - while reading a news item, add a stock to your watchlist on icidirect
 - request hotel/travel reservations for an event
 - order a print from your photo/t-shirt/greeting card printing service by right clicking on any image you see on the web
 - search for jobs on naukri.com at a company mentioned in some blog/article
- Micropayments

Technical reasons to use REST

- Leverages existing web infrastructure for scaling e.g. load distribution using URLs, exploiting proxy caches, security, compression
- Versioning is easier
- One step closer towards interoperability
- Lighter weight
- Enables architectures that don't have a presentation layer on the server side.

ThoughtWorks' Fight-Bloat Manifesto

- Lean/Agile methods *over* heavyweight processes
- Lightweight containers *over* EJB-like components
- Terse dynamic languages *over* verbose static languages
- RESTful web services *over* WS-* services

REST is yet another step along the path to revolutionize IT.

Comments? Questions?

Thank you

To get started with this blank [TiddlyWiki](#), you'll need to modify the following tiddlers:

- **SiteTitle** & **SiteSubtitle**: The title and subtitle of the site, as shown above (after saving, they will also appear in the browser title bar)
- **MainMenu**: The menu (usually on the left)
- **DefaultTiddlers**: Contains the names of the tiddlers that you want to appear when the [TiddlyWiki](#) is opened

You'll also need to enter your username for signing your edits: