

Intelligent Web Applications

from *data* to *services*: the *programmable* web

Eyal Oren

Vrije Universiteit Amsterdam

4 September 2008

Mashup Dashboard - ProgrammableWeb

http://www.programmableweb.com/mashups

Sign up/Sign In Add a Link Subscribe Tag Cloud Google Custom Search Advertise on PW Contact Us Press About FAQ

programmableweb

Need Business Mashups?
JUST @#\$% IT! SERENA

Home News Mashups APIs Verticals How-To Contests Members

Dashboard Directory Newest Most Popular Mashup Matrix Tag Cloud Random

Subscribe
5 All New Mashups

Mashup Directory
Total Mashups Listed
3077

Past 7 Days: 21
Past 30 Days: 87

Mashups/Day
3.13

7 Days Avg.: 3.00
30 Days Avg.: 2.90

Our Sponsors
Got maps? Get cash. lat49

Popular Directory Searches
Celebrity Mashups
Video Mashups
Popular New Mashups

Mashup Dashboard
Thousands of web mashups with new updates daily

Mashup of the Day, May 29, 2008

TwitterWheel
Find out which of your Twitter friends know each other.
APIs: Twitter
Tags: fun, messaging, microblogging

★★★★★

All Mashups | Popular Mashups | Matrix | Add Yours | How-To Guide

Top Mashup Tags

Last 14 Days All

- mapping (25%)
- messaging (15%)
- social (13%)
- shopping (6%)
- search (6%)
- video (6%)
- microblogging (6%)
- bookmarks (6%)
- blog (5%)
- photo (5%)

ProgrammableWeb Sponsors

BT Web21CSDK Do Less : Achieve More web21c.bt.com

napLogic Enabling the Mashable Enterprise OPEN SOURCE GET IT TODAY

thumbplay ADD MOBILE TO APPS. GET PAID FOR YOUR WORK. CLICK HERE

openkapow Mashups in Minutes No Kidding!

Get apps. Get paid. Money

O'REILLY GRAPHING SOCIAL PATTERNS Userplane Washington, DC June 9-11, 2008

Web 2.0 for business

STRIKEIRON Build Something. Over 100 Web Services

Done

programmableweb.com

wrije Universiteit amsterdam JSSS

Goals

- ▶ integration requirements: data structure, messages, operations
- ▶ the REST principles: promote scalability
- ▶ the WS standards: abstract implementation heterogeneity
- ▶ how some service standards violate REST principles
- ▶ how some service scenarios require additional standards
- ▶ freedom from choice vs. freedom of choice

How to participate

- ▶ Listen, take notes, ask questions
- ▶ Read the material, think about it, correct me
- ▶ Reading material, on Blackboard
 - ▶ Erenkrantz et al.: Web architectures (Sect.1–5)
 - ▶ Gottschalk et al.: Web services
 - ▶ Pautasso et al.: REST vs WSDL
 - ▶ Fielding and Taylor: the REST style
- ▶ Exercise: service mashup, next week

Data on the Web

- ▶ HTML: Web documents
- ▶ CSS: reusable layouts
- ▶ XML: arbitrary Web data
- ▶ XSD/DTD: XML schemas
- ▶ XQuery/Xpath: accessing XML data
- ▶ XSLT: transforming XML data

today

Services on the Web

- ▶ Reuse data, reuse functionality, reuse services
- ▶ Screen-scraping, custom protocols, Web services
- ▶ Service: software component at network-accessible endpoint
- ▶ Web service: service accessible using common Web standards
 - ▶ “Big” services: SOAP, WSDL, UDDI
 - ▶ “Light” services: REST, HTTP, JSON
- ▶ Mashups: Google maps, Facebook widgets

Software as a Service

The "Killer Apps" of the New Millennium

The slide features a collection of logos for prominent web services and products. On the left, a vertical red bar contains the number '16'. The logos are arranged in a grid-like fashion. At the bottom right, the O'Reilly logo is displayed, consisting of the word 'OREILLY' in a bold, sans-serif font above a stylized bird logo and the name 'O'Reilly' in a script font.

16

OREILLY[®]
O'Reilly

Software as a Service

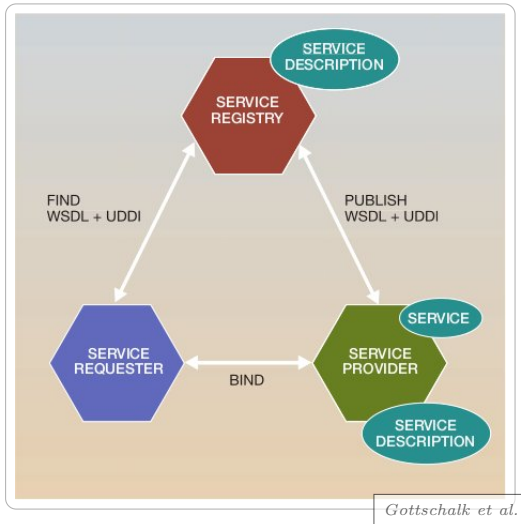
What Makes Them Interesting To Me

- The Internet, not the PC, is their platform
- Built on top of open source, but not themselves open source
- Services, not packaged applications
- Exploring how to become platform players via web services APIs
- Data aggregators, not just software
- Network effects from user contributions key to market dominance
- The most successful are “semantic learning systems”, leveraging implicit metadata

OREILLY®

O'Reilly

Service-oriented architecture



Service-oriented architecture

- ▶ Abstract implementation heterogeneity
- ▶ Abstract platform heterogeneity
- ▶ No shared objects (no shared state)
- ▶ Standardise data structure and message protocol
- ▶ (Standardise service description and discovery)

REST: Representational State Transfer

- ▶ Architectural style: servers, clients, proxies, caches
- ▶ Web: set of hyperlinked resources
- ▶ Resources: items identified by URIs
- ▶ Accessing resources: returns some *representation* (eg. HTML)
 - ▶ Representation puts client in some state
 - ▶ Traversing hyperlinks: accessing another resource
 - ▶ Traversing hyperlinks changes client's state
- ▶ Web application: state machine
 - ▶ states: resource representations
 - ▶ transitions: hyperlink traversal
- ▶ Captures success of the Web, guides its evolution
- ▶ Many Web services are RESTful (by design or not)

[Singh and Huhns, *Service-Oriented Computing*, 2005]

Why is state a bottleneck for load-balancing?

Imagine shopping at Albert Heijn: don't carry your shopping cart around, and check-out only at your personal cash point

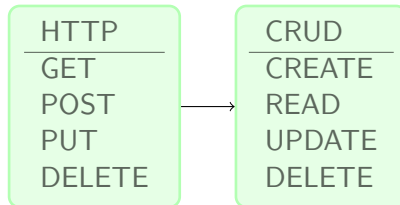
REST principles

- ▶ Information abstracted into resources, identified by URIs
- ▶ Resource representation: bytes and representation metadata
- ▶ All interactions are context-free
- ▶ Only few primitive operations
- ▶ Caching through idempotent operations and metadata (independent of resources)
- ▶ Intermediaries are encouraged

REST and AJAX

- ▶ Asynchronous resource acquisition, client-side javascript
- ▶ Content is interpreted by client, moves computation to client
- ▶ REST reduces server-side state load
- ▶ AJAX reduces server-side computational load

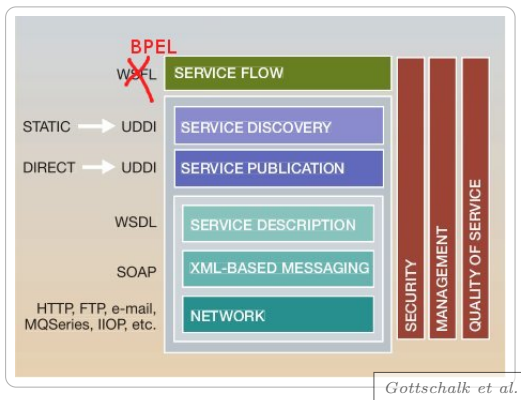
Operations in REST



Web service standards

- ▶ message transfer (SOAP, HTTP)
- ▶ data structure (XML, XSD)
- ▶ interface description (WSDL/text)
- ▶ service discovery (UDDI/hyperlinks)
- ▶ security (HTTPS/WS-Security)
- ▶ reliability (WS-Reliability)
- ▶ transactions (WS-Transactions)

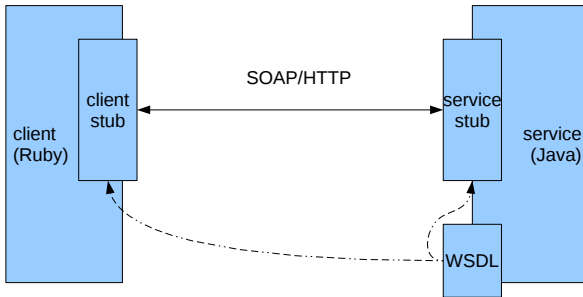
Web service standards



WSDL services

- ▶ Web Service Definition Language
- ▶ Message: XML Schema data structure
- ▶ Operation: pattern of input/output messages
- ▶ Interface (portType): set of operations
- ▶ Binding: interface with network protocol
- ▶ Service: binding at concrete endpoint (port)

Invoking Web services



How SOAP violates REST

- ▶ Uses mostly *POST operations*, with operation semantics in message payload: intermediaries cannot cache without knowing receiver's operation semantics
- ▶ Wraps *SOAP envelope* inside HTTP bodies, SOAP metadata not exposed as HTTP metadata

Summary

- ▶ integration requirements: data structure, messages, operations
- ▶ the REST principles: promote scalability
- ▶ the WS standards: abstract implementation heterogeneity
- ▶ how some service standards violate REST principles
- ▶ how some service scenarios require additional standards
- ▶ freedom from choice vs. freedom of choice