Web Services

Lorna Mitchell
Freelance PHP consultant
@lornajane
http://lornajane.net

"I see the world from the command line."

What is a "web service?"

- Expose information
- Expose functionality
- Client = machine
- Create clean HTTP boundaries
- Enable separate scaling

Use curl – eliminates points of failure

http://www.lornajane.net/posts/2008/Curl-Cheat-Sheet

Every service should have a heartbeat

- flickr.test.echo
- Echo passed params
- Stop idiot DOS reports

Every service should have documentation, (real) examples, and a support mechanism

HTTP

- Use the headers
  - Accept and Content-Type – content format negotiating – pay attention
    - text/html accept – useful for debugging
    - Support multiple formats!
    - Parse prioritized list (http://arbitracker.org – src/classes/request/parser.php)
  - User-Agent – what made the request? Tailor response to client
  - Set-Cookie and Cookie – working with cookie data – can be a nice addition to a service
- Use the status codes
  - 200 OK
  - 302 Found
  - 301 Moved
  - 401 Not Authorized – confused w/403 – "I don't know who you are"
  - 403 Forbidden – "I know who you are and am NOT letting you in"
  - 404 Not Found

    - 500 Internal Server Error – “Go read your logs”
- Use the verbs
    - GET - read
    - POST - create
    - PUT - update
    - DELETE – delete

Give consumers a choice of formats

- Detect header, parameter override
- JSON, XML, HTML, ?

Include a version parameter

Statelessness

- BAD: toggle (joindin API “I'm attending”)
- Self-contained, complete, requests
- Repeatable and predictable

Consuming from PHP

- file_get_contents – GET requests
- CURL (yuck)
- Pecl_HTTP (yum)

Service Types

- RESTful
    - Multiple endpoints (URL for each method)
    - HTTP verbs indicate the operation
    - More religious zealots (“HTTP Web Service” vs “RESTful Service”)
    - Typically supports CRUD operations on multiple entities
        - /user/add
        - /user/3/profile
        - /user/3/delete
        - …
    - Hypermedia – providing links to related items/collections
        - Can change URLs easily
        - Self-documenting
    - Create: POST: HTTP 201 Created + Location header / HTTP 400 if create fails
    - Update: PUT: HTTP 204 OK / HTTP 400 if update fails
        - parse_str(file_get_contents('php://input'), $data);
    - Read: GET: HTTP 200 or 302 (if moved) / HTTP 404 if not found
    - Delete: DELETE: HTTP 200 (always)

- RPC
  - XML-RPC (Flickr), JSON-RPC
    - Single endpoint
    - Method names
    - Method params
    - Return value
    - “RPC that uses XML” vs true “XML-RPC” (cf. Wikipedia)
  - SOAP
    - Subset of XML-RPC
    - Optional WSDL (complicated)
      - Don't write by hand
      - Can be generated from PHPDoc comments
      - Read backwards
        1. Namespaces
        2. Definitions
        3. Data types
        4. Functions
        5. Endpoint
    - Easy to publish own SOAP service from a PHP class (PHP Soap libs well-written)
    - Response is the same as calling the class locally
    - Hard to debug
      - trace=1 in the options
      - getLastRequest*, getLastResponse* methods
      - Wireshark
      - Charles Proxy (can help with debugging over https)
      - CURL
      - SoapUI (http://soapui.org)
    - Well-known
    - Well-supported in other langs
    - Very verbose on data formats which can cause cross-language problems

Make your service as good as possible
- Naming conventions
- Parameter validation
- Parameter order consistency

Access Control
- Username/password with every request – only over https
- Login action, token – just like normal sessions
- OAuth

Error Handling defines API quality
- “Golden rule”: use expected response format for errors (don't croak in HTML if the user asked

for JSON or XML)
- Bundle multiple errors (esp. parameter validation errors)
- Give helpful error messages
- Be consistent

Reliability is key
- Unit testing
- Source control
- Heartbeat monitoring
- Automated deployment

Documentation
- You can never know who in advance your users will be
- Provide a quick-start for the impatient
- Provide real-life working examples

http://www.slideshare.net/lornajane/web-services-tutorial
http://bit.ly/emRpYT
http://joind.in/talk/view/3338

Frontend Caching

Helgi

Pareto Principle (80/20 rule)
20% of the effort produced 80% of the results

80% of response time is spent downloading resources

Rules of Web Performance

1. Weight
2. Time
3. Processing
4. Perception
    - Responsiveness
    - Amazon
    - Make people **think** your site is fast
    - 50% of users arrive with an empty cache

Cookies
- Cookies are sent with static content – huge cookies are bad for performance

- Upload speeds are much slower than download speeds
- Limit cookie usage to domains that need them
- Expire cookies

Parallel Downloads
- Browsers have per-domain concurrent download limits
- IE6/7 – 2 concurrent downloads max (browserscope.org)
- Use multiple static domains (CNAME records)
- Static domains don't use cookies
- Can cause problems if overdone (DNS thrashing – CPU spikes)
- 2-4 subdomains is a good average

Combine Files (judiciously)
- Challenge – dev separate modules
- Challenge – loading more than needed

Javascript
- Breaks parallel download rules
- **script defer** adversely affects older versions of Firefox

Lazy Loading
- Load above the fold first

Minify Javascript and CSS
- Javascript
  - UglifyJS (uses Node.JS)
  - Google Closure
  - YUI Compressor
  - Dogo Shrinksafe
  - JSMin
- CSS
  - YUI Compressor
  - minifycss.com
  - OOCSS

Gzip compression
- Works well on any text data
- Don't use on binary files

Save HTTP 404 bandwidth
- robots.txt
- favicon

Images
- Don't oversize favicon
- Badly optimized
- Don't cheat on thumbnails

- Better compression tools
  - OptiPNG
  - pngwolf
  - pngcrush
  - jpegtran

CSS Sprites
- Combine images
- Use CSS positioning
- Hard to maintain long-term

Resource Packages
- JAR file that contains everything
- Standardization WIP

SSL
- Latest OpenSSL has Google performance patches

Test with slower connections

Use a CDN

Reverse Proxies
- Alternative to etags for server farms
- Varnish, Nginx, Squid
- http://bit.ly/query_rev_comp

Tools
- Firebug
- Yslow
- PageSpeed
- Chrome Inspector
- HTTPWatch.com
- WebPageTest.com
- HTTPArchive.org
- Yottaa.com
- WonderProxy.com
- pagespeed.googlelabs.com
- BrowserScope.com
- html5boilerplate.com
- Diffable (http://code.google.com/p/diffable/)
- mod_pagespeed