



Why SOAP Sucks and Why SOAP Rocks

Matt Sergeant



Why SOAP Sucks



Implementation Issues

- SOAP is hard to implement *fully*
- Most implementations only implement a subset of W3C XML Protocols
- Technologies are:
 - SOAP
 - WSDL
 - UDDI
 - XML Schemas
 - ◆ *data typing*



Logging

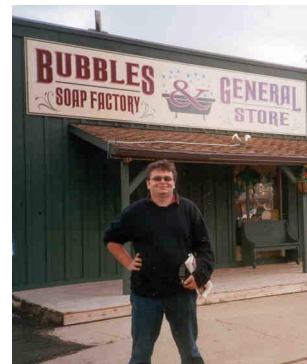
- SOAP has no built-in logging
- Need to write our own custom logging functions
- Each method needs to do its own logging
- No standard logging API means each application does logging differently

Security

- SOAP goes via Single Endpoint

POST http://server/soapserver.pl HTTP/1.0

- No security model (yet)
 - ◆ .NET vs Sun ONE vs ?
 - ◆ Security model being worked on at the W3C
- Complex Security issues, as with any RMI system
- Bypasses firewall security measures





Session Persistence

- SOAP has no well defined session model
 - ◆ Contrast this with the web and cookies with a session id
- SOAP has no cookies
- Requires Flyweight objects - objects that are just ID values
- Plus requires support in your SOAP server for garbage collection
- None of this scales like the web does

POST vs GET

- SOAP uses HTTP POST everywhere
- This bypasses server and proxy based caching
- It also make it hard to build web-like services on top - like search
- POST therefore loses us some scalability
- POST is also invalid for a lot of requests being made
 - ◆ e.g. we POST to our SOAP server to *get* the stock quotes
 - ◆ POST implies modification, GET implies non-modification
- HTTP offers other verbs too - PUT, DELETE, etc

Human Readable

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.stock.org/stock" />
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>
  </soap:Body>
</soap:Envelope>
```

- Ahem.



Methods aren't enough

- Developers appear to need APIs
- Case in point: Google
- If we have to ship APIs anyway, we don't need SOAP



REST

No, it's not break time ;-)

Representational State Transfer

- An awful mouthful
- Describes the web as we know it
- Using HTTP to the max - for applications rather than web pages
- Uses the vocabulary of HTTP rather than building a new vocabulary
 - ◆ POST to modify resources
 - ◆ PUT to store new resources
 - ◆ GET to retrieve resources
 - ◆ DELETE to delete resources



REST Examples

- Stock Quotes:

```
$ lwp-request http://server/getstockquote.pl?symbol=MSFT  
51.11
```

- Web Search

```
$ lwp-request http://giggle.com/search?q=axkit;xml=1  
<results>  
  <hits>66,900</hits>  
  <time-taken>0.19</time-taken>  
  <result href="http://axkit.org/">  
    <title>Apache AxKit</title>  
    <description>XML Application Server for Apache. Provides  
    on-the-fly conversion from XML to any format, such as...</description>  
    <category href="http://directory.google.com/Top/Computers/Software/Internet/Servers/A  
    >Computers > Software > Internet > Servers > Application > Open Source</category>  
  </result>  
</result>  
  ...
```



REST Benefits

- Works *with* your HTTPD
- Logging comes for free
- Authorisation/Authentication comes for free
- GET/POST can be used effectively, and cache friendly
- Persistent sessions well understood



Why SOAP Rocks

SOAP::Lite

- Simplest module for SOAP in any language
- Just plain works
- Server:

```
use SOAP::Transport::HTTP;  
SOAP::Transport::HTTP::CGI->dispatch_to('Module::Name')->handle;
```

- Client:

```
use SOAP::Lite +autodispatch =>  
  uri => 'http://www.soaplite.com/Temperatures',  
  proxy => 'http://services.soaplite.com/temper.cgi';  
  
my $temperatures = Temperatures->new(32); # get object  
print $temperatures->as_celsius;          # invoke method
```



Developer Ease

- Modules available for almost all languages
- Frameworks like .NET make building SOAP servers and clients trivial
- Don't have to understand HTTP issues to build SOAP services



Interoperability + Multiple Implementations

- Most clients interoperate with most servers
- Mostly SOAP tends to "just work", unless you have complex needs
- More than 29 implementations

W3C Standard

- This seems to carry more weight than the IETF these days
- Easily accessible information
- Future development of the standard is certain



REST Downsides

- No standard - REST means everyone designs their own application
 - ◆ While this has worked well for web browsers, its a lot harder for computers to adapt
- No REST tools beyond httpd + server languages
- Sniffers can't comprehend REST like they will be able to comprehend SOAP

Conclusions?

- As always, the application will dictate the method
- Or pointy haired bosses
- SOAP sucks
- SOAP *implementations* rock
- REST is better if you can use it
- Thank You.

