Microsoft Exchange and SOAP

(my two great loves)

Oliver Gorwits

miltonkeynes.pm Tuesday 12th October 2010

All I want for Xmas...

- Is to speak to our new MS Exchange 2007
- To retrieve my Calendar so my wife can see when I'm busy
- MS Exchange 2007 ships with **Exchange Web Services** - a SOAP API

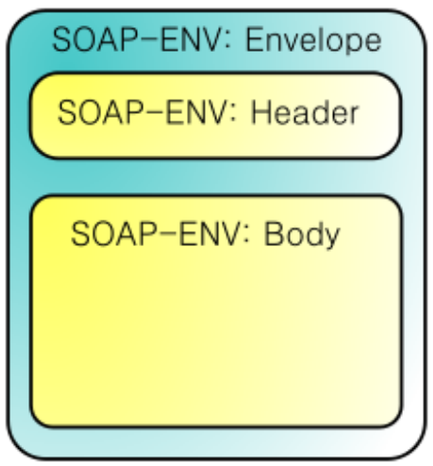
An API - easy, right?

- SOAP is horrific to the uninitiated
- But it has some definite positives
- Most REST/RPC is not **described**
  - This leads to mistakes and errors
- SOAP is XML, and describable (Schema)

SOAP Messages

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:GetStockPrice xmlns:m="http://www.example.org/stock">
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```



WSDL and Schema

- Describes the messages
  - That is: calls, arguments, data types, data structures, responses, etc
- Fairly Dense, all specified in XML

WSDL Example

```
<types>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
               xmlns="http://www.tmsws.com/wsdl20sample"
               targetNamespace="http://www.example.com/wsdl20sample">

       <xs:element name="request">
          <xs:complexType>
             <xs:sequence>
                <xs:element name="header" maxOccurs="unbounded">
                   <xs:complexType>
                      <xs:simpleContent>
                         <xs:extension base="xs:string">
                            <xs:attribute name="name" type="xs:string" use="required"/>
                         </xs:extension>
                      </xs:simpleContent>
                   </xs:complexType>
                </xs:element>
                <xs:element name="body" type="xs:anyType" minOccurs="0"/>
             </xs:sequence>
             <xs:attribute name="method" type="xs:string" use="required"/>
             <xs:attribute name="uri" type="xs:anyURI" use="required"/>
          </xs:complexType>
       </xs:element>
```

SOAP::Lite

- Doesn't know much about WSDL
- You have to do a lot by hand
- Hence know about SOAP and its family
- Data type handling seems to be fragile

XML::Compile::*

- Mark Overmeer
- Huge, complex, but works really well
  - A bit like SOAP, then
- Perl data structures in and out
- Handles Fault Messages
- So easy even I could use it

XML::Compile::*

```
# preparation
use XML::Compile::WSDL11;      # use WSDL version 1.1
use XML::Compile::SOAP11;      # use SOAP version 1.1
use XML::Compile::Transport::SOAPHTTP;

my $wsdl = XML::Compile::WSDL11->new($wsdlfile);
$wsdl->addWSDL(...more WSDL files...);
$wsdl->importDefinitions(...more schemas...);
```

```
# during initiation, for each used call (slow)
my $call = $wsdl->compileClient('GetStockPrice', ...);

# at "run-time", call as often as you want (fast)
my $answer = $call->(%request);

# capture useful trace information
my ($answer, $trace) = $call->(%request);

# investigate the %request structure (server input)
print $wsdl->explain('GetStockPrice', PERL => 'INPUT');

# investigate the $answer structure (server output)
print $wsdl->explain('GetStockPrice', PERL => 'OUTPUT');
```

EWS::Client and EWS::Calendar::Viewer

```
use EWS::Client;
use DateTime;

my $ews = EWS::Client->new({
    server      => 'exchangeserver.example.com',
    username    => 'oliver',
});
```

```
my $entries = $ews->calendar->retrieve({
    start => DateTime->now(),
    end  => DateTime->now->add( month => 1 ),
});

print “I retrieved “. $entries->count .” items\n”;

while ($entries->has_next) {
    print $entries->next->Subject, “\n”;
}

my $contacts = $ews->contacts->retrieve;
```

Fin