



Web Services Introduction

Walden Leverich
Tech Software
WaldenL@techsoftinc.com

What's a Web Service?



- W3C says a web service:
 - “a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.”
- Isn't that clear?

What's the problem?



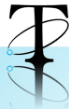
- Disparate systems that need to talk to each other!
 - Batch doesn't cut it.
 - Real-time integration w/partners
 - Separation of concerns w/in an organization
- Not the first attempt at solving the problem
 - Sun's RMI
 - Microsoft's DCOM
 - OMG's CORBA
 - EDI
 - ebXML
- Not only B2B
 - Used w/in a website for Ajax communications

What's a Web Service, really?



- Simple! Program A calls Program B
 - but it happens over a network
 - and it doesn't matter what language they're written in
 - nor what platform they're running on
 - and the network can be the internet
 - and it can be secure
 - and reliable
 - and fast
 - and ...

Whoa! SOA?



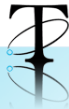
- SOA = Service Oriented Architecture
- Aren't Web Services all about SOA
 - Isn't SOA all about Web Services?
 - Not necessarily!
- SOA is great industry buzzword for enterprise integration
 - IBM, Sun, BEA, HP, Software AG, Oracle, Microsoft, etc. are all offering "SOA Solutions"
- Key part of SOA is the word Architecture
 - SOA isn't a "plug in", it's a way of building systems in the enterprise
 - Does push the concept of reuse, but it's complex
 - Not needed to take advantage of Web Services

HTTP



- Hyper Text Transfer Protocol
 - Not just text!
 - Same protocol for JPG, PDF, Flash, JS, CSS, etc.
- Standard for addressing web pages, media, *services* and other resources
- Address is a URL, or Uniform Resource Locator
 - http://www.cnn.com
 - https://www.mycompany.com/services/CheckInv.php
- A URL is a special case of a URI
 - Uniform Resource Identifier

XML



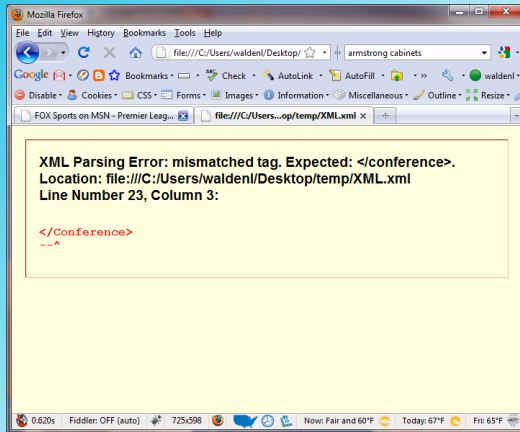
- eXtensible Markup Language
 - Encoding of "message"
- Strong Unicode support for national languages
- Markup Language used to format data for transfer between service consumer (caller) and provider (callee)
- Descendant of SGML
- Not meant to be non-programmer readable!
 - Barely programmer readable
- Modern HTML is XML by definition (XHTML)

XML Details

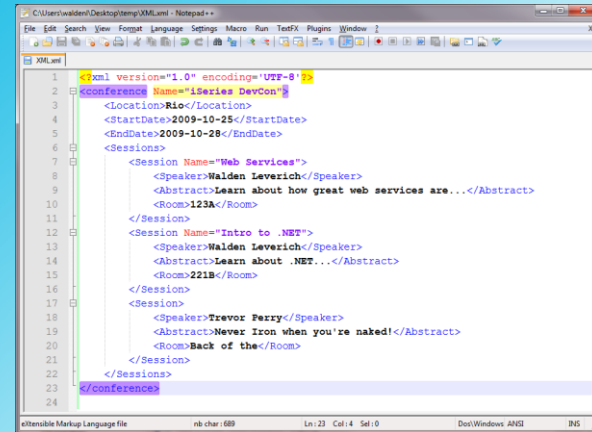


```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <conference Name="iseries DevCon">
3   <Location>Rio</Location>
4   <StartDate>2009-10-25</StartDate>
5   <EndDate>2009-10-28</EndDate>
6   <Sessions>
7     <Session Name="Web Services">
8       <Speaker>Walden Levezich</Speaker>
9       <Abstract>Learn about how great web services are...</Abstract>
10      <Room>123A</Room>
11    </Session>
12    <Session Name="Intro to .NET">
13      <Speaker>Walden Levezich</Speaker>
14      <Abstract>Learn about .NET...</Abstract>
15      <Room>221B</Room>
16    </Session>
17  </Sessions>
18  <Session>
19    <Speaker>Trevor Ferry</Speaker>
20    <Abstract>Never Iron when you're naked!</Abstract>
21    <Room>Back of the</Room>
22  </Session>
23 </Conference>
24
```

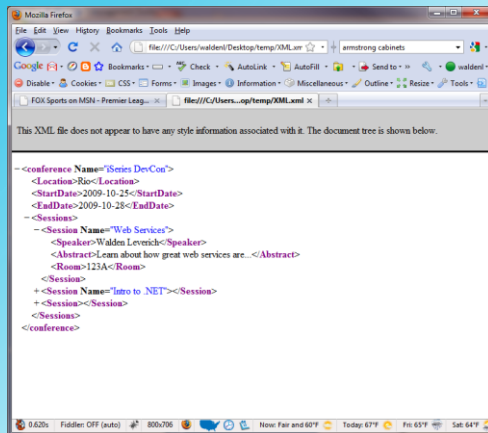
Ooops! XML is case sensitive!



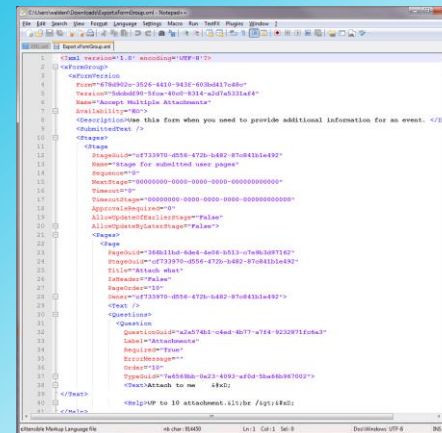
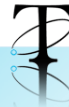
Fix the end tag!



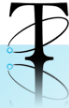
Correctly Formatted XML



Production XML is more complex!



SOAP



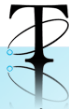
- Simple Object Access Protocol
 - Sending encoded messages
- Standard for describing messages passed between programs using XML
 - “Message Passing” == Program call
- SOAP used to request service from another program, and pass it parameters and get parameters back
- And you don’t actually have to use HTTP, you could use SMTP, Sockets, Carrier Pigeons, SOAP handles it all

WSDL



- Web Services Description Language
 - The “how to” manual for a web service call
- Compile-time artifact in some cases
 - Needed to determine the rules of the service call, not necessarily needed at runtime
- Run-time artifact in other cases
 - Eg. PHP, Ruby
- XML based standard describing the web service parameters and how to call it
 - XML describing the XML you need to send – circular!
- Machine-readable “documentation” for a service

WSDL Parts



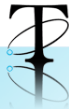
- WSDL made up of 5 parts, or “elements”
 - Types, Messages, Ports and Operations, Bindings, and Service
- Service describes the web service at a high level
 - What does the service do
 - How can I access it? (Ports)
 - Service analogous to service program
- Ports describe the access points, ports of call
 - SOAP RPC, GET, POST are common
- Bindings describe how the data is encoded for each port
 - Literal, MIME and URLEncoded are common

WSDL Parts 2



- Operations describe the different methods available in a service
 - Single service can provide many methods, typically related
 - GetPrice, SetPrice, CheckInventory, CheckOrder, CreateOrder...
 - Analogous to procedures in service program
- Types defines the details of a parameter
 - Types analogous to a data-structure
 - Subfields in a DS can be other data structures, similarly, Types can (typically do) reference other types

Big vs Little Web Services



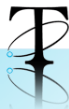
- SOAP and WS-* vs. RESTful
 - Two camps in web services
- SOAP camp is used for back-end enterprise services
 - Required for many Enterprise Service Buses (ESBs)
 - Requires for WS-* spec
- REST camp is used as “web APIs”
 - Simpler to call from web browsers
 - Basis for many Ajax based sites

WS-* What?



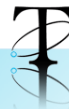
- WS-I – Web Services Interoperability Organization
 - Publishes “Profiles” or guidelines for web service interoperability
 - Basic Profile, 1.1, 1.2, 2.0, etc.
 - Industry consortium
 - IBM, Microsoft, BEA, SAP, Oracle, Fujitsu, HP, Intel, Sun, et al
- WS-* Standards
 - WS-Security
 - WS-Reliability
 - WS-Transaction
 - WS-Addressing

WS-Security



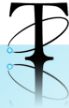
- Um, aren’t web services secure?
 - Define “secure”
- Describes how integrity and confidentiality can be ensured in a web service call
 - SAML (Security Assertion Markup Language)
 - Kerberos
 - X.509 (Public Key Infrastructure, and Single Sign-on)
- Non-repudiation
 - Signatures and encryption
 - The caller claims to be Bob. How can I guarantee that it is?

WS-Reliability



- Um, aren’t web services Reliable?
 - Again, define “Reliable”
 - Is your i5/OS app “reliable?”
- Ensures message delivery and integrity
 - System failures
 - Network failures
 - Software failures
- Delivery Assurances
 - At Least Once – Could be 3 times
 - At Most Once – Could be 0 times
 - Exactly Once
 - “Give me a ping, Vasili. One ping only, please.”
 - In Order – FIFO, plus one of the above

WS-Transaction



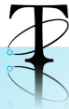
- WS-Atomic Transaction
 - Commitment control for web service messages
 - All or nothing transaction between multiple services
- WS-Business Activity
 - Long running business process
 - Umbrella for many Atomic transactions
 - Can wait on Human elements
 - All state transactions are recorded

What happened to *simple*?



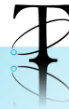
- Remember SOAP was *Simple* Object Access Protocol?
 - WS-* is optional
 - WS-* is mostly ignored
 - But when you need it you really need it
- Tooling makes SOAP web services a breeze
 - Really, it does
 - ...if you don't need too much complication
- This is way too painful, I need a rest!

Ah good, my REST!



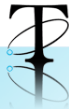
- Representational State Transfer
 - Simpler web service call architecture
- Alternative to SOAP
 - But doesn't support WS-* (Yippee!)
- Based on the fundamental verbs in HTTP itself
 - Doesn't need (or support) a WSDL concept for describing the service – human documentation only
- Much simpler to consume a REST service from a browser
 - Ideal for Ajax
- Responses don't have to be XML
 - In fact, often aren't – that's the point!

REST Architecture



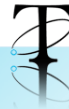
- Client-Server
 - Separation of concerns
 - Clients don't care about storage
 - Servers don't care about state or user interface
 - Increased reuse and portability
 - Better scalability and reliability
- Stateless
 - No state stored on server
 - Client remembers all state
 - Each request contains everything server needs to know

REST Architecture 2



- Cacheable
 - Clients and proxies are allowed to cache results
 - Same result for same request
 - Increased performance, and edge caching possible
- Uniform interface
 - Clients and Servers code to common interface
 - Each can develop on their own details
- Layered system
 - Client doesn't care if it's talking to end server, or an intermediary, as long as it gets its answer
- Code on demand
 - Servers can push code (Javascript) to client to run on client and extend client capabilities

REST Guiding Principles



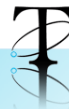
- Identification of Resource
 - Each resource is uniquely identified by a URI
 - Conceptually separate from the server representation
 - You get a customer representation, not a customer row from the database
- Manipulation of Resources
 - URI contains sufficient information to manipulate (create, change, delete) server resource – with permission

REST Guiding Principles 2



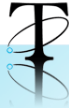
- Self-Descriptive
 - There is no WSDL
 - Media-type (MIME/Internet Media Type) must provide sufficient information for processing
 - Shouldn't need to look inside message
- Hypermedia is engine of state
 - Likely to be related resources
 - Should be identified in response complete with URIs for accessing them

REST Basics



- Given: We'll use HTTP to make service calls
- Then let's use ALL of HTTP
- HTTP is a text based protocol based on verbs
 - Verbs are commands sent from client to server
- GET and POST are most common, they handle 99% of the web
- But there are other verbs in the core HTTP protocol
 - HEAD, PUT, PROPS, DELETE, TRACE, OPTIONS, etc.
 - Rare, but you can define your own verbs too.

Typical Web Services



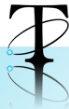
- Outside vendor information
 - Checking Stock availability real-time
- Internal Cross-System Services
 - Public website on IIS getting data from iSeries
- Sales Tax Calculations
 - Outsource the problem of keeping tables updated
- UPS/FedEx Shipping Info
 - Lookup current shipment status to keep customers informed
- Weather reports
 - Delivery scheduling or activity management based on weather

More Web Services



- Currency Exchange
 - Show customers price in their currency at current rates
- Postal Address Validation
 - Check and normalize an address for delivery
- IP Address Information
 - Requestor in the USA? On dialup? In China?
 - Geographic information serving – closest stores/dealers
- Email Address
 - Correctly Formatted?
 - On a “free” email service (gmail, hotmail, etc)
 - Real address?
 - me@company.com; nowhere@nowhere.com

Consumers and Providers



- Program A calls Program B
 - Program A is the consumer
 - Program B is the provider
- Getting or sending data somewhere?
 - You’re a consumer
- Providing or accepting data from somewhere?
 - You’re a provider
- Most of the time you’re a consumer
 - Good! It’s the simpler of the two!

IP Address Validation



Getting IP Address Info



- DOTS IP Address Validation
 - Given an IP Address return information about it
 - <http://www.serviceobjects.com/products/ip-address>
- Service Endpoint
 - http://trial.serviceobjects.com/gpp/GeoPinPoint.asmx/GetLocationByIP_V2
- Parameters IN on URL (the "GET" method)
 - IP Address to lookup
 - LicenseKey (How they handle security)
- XML Out

Reading the WSDL



- In SOAP service the definition is contained in WSDL
- Documentation says WSDL available at
 - <http://trial.serviceobjects.com/gpp/GeoPinPoint.asmx?WSDL>
- From WSDL we look for
 - Ports and bindings available, we want the GET one
 - Operations available on that Port, We want GetLocationByIP_V2
 - Messages available (we want the IP Lookup one)
 - Parameters expected
 - Parameter passing method

Service and Ports



```
209 </wsi:operation>
210 <wsi:operation name="GetLocationByIP_V2">
211   <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Returns geographic location and pro:
212 </documentation>
213 <wsi:input message="tns:GetLocationByIP_V2HttpPostIn" />
214 <wsi:output message="tns:GetLocationByIP_V2HttpPostOut" />
215 </wsi:operation>
216 </wsi:portType>
217 <wsi:binding name="DOTSGeoPinPointSoap" type="tns:DOTSGeoPinPointSoap">
218 </wsi:binding>
219 <wsi:binding name="DOTSGeoPinPointHttpGet" type="tns:DOTSGeoPinPointHttpGet">
220 </wsi:binding>
221 <wsi:binding name="DOTSGeoPinPointHttpPost" type="tns:DOTSGeoPinPointHttpPost">
222 </wsi:binding>
223 <wsi:service name="DOTSGeoPinPoint">
224 <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">For more information on our web servi
225 </documentation>
226 <wsi:port name="DOTSGeoPinPointSoap" binding="tns:DOTSGeoPinPointSoap">
227 <soap:address location="http://trial.serviceobjects.com/gpp/GeoPinPoint.asmx" />
228 </wsi:port>
229 <wsi:port name="DOTSGeoPinPointHttpGet" binding="tns:DOTSGeoPinPointHttpGet">
230 <http:address location="http://trial.serviceobjects.com/gpp/GeoPinPoint.asmx" />
231 </wsi:port>
232 <wsi:port name="DOTSGeoPinPointHttpPost" binding="tns:DOTSGeoPinPointHttpPost">
233 <http:address location="http://trial.serviceobjects.com/gpp/GeoPinPoint.asmx" />
234 </wsi:port>
235 </wsi:service>
236 </wsi:definitions>
```

Bindings



```
246 <wsi:binding name="DOTSGeoPinPointHttpGet" type="tns:DOTSGeoPinPointHttpGet">
247 <wsi:binding xmlns="http://schemas.xmlsoap.org/wsdl/" />
248 </wsi:binding>
249 <wsi:operation name="GetLocationByIP">
250 <http:operation location="GetLocationByIP" />
251 <wsi:input>
252 <http:urlEncoded />
253 </wsi:input>
254 <wsi:output>
255 <mime:mimeXml part="Body" />
256 </wsi:output>
257 </wsi:operation>
258 <wsi:operation name="GetCountryByIP">
259 <http:operation location="GetCountryByIP" />
260 <wsi:input>
261 <http:urlEncoded />
262 </wsi:input>
263 <wsi:output>
264 <mime:mimeXml part="Body" />
265 </wsi:output>
266 </wsi:operation>
267 <wsi:operation name="GetLocationByIP_V2">
268 <http:operation location="GetLocationByIP_V2" />
269 <wsi:input>
270 <http:urlEncoded />
271 </wsi:input>
272 <wsi:output>
273 <mime:mimeXml part="Body" />
274 </wsi:output>
275 </wsi:operation>
276 </wsi:binding>
```

Port Types

```

182 <wsdl:portType name="DOTSGeoPinPointHttpGet">
183   <wsdl:operation name="GetLocationByIP">
184     <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Returns geogra
185     <wsdl:input message="tns:GetLocationByIPHttpGetIn" />
186     <wsdl:output message="tns:GetLocationByIPHttpGetOut" />
187   </wsdl:operation>
188   <wsdl:operation name="GetCountryByIP">
189     <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Returns the co
190     <wsdl:input message="tns:GetCountryByIPHttpGetIn" />
191     <wsdl:output message="tns:GetCountryByIPHttpGetOut" />
192   </wsdl:operation>
193   <wsdl:operation name="GetLocationByIP_V2">
194     <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Returns geogra
195     <wsdl:input message="tns:GetLocationByIP_V2HttpGetIn" />
196     <wsdl:output message="tns:GetLocationByIP_V2HttpGetOut" />
197   </wsdl:operation>
198 </wsdl:portType>
  
```

Messages

```

158 <wsdl:message name="GetLocationByIP_V2HttpPostIn">
159   <wsdl:part name="IPAddress" type="s:string" />
160   <wsdl:part name="LicenseKey" type="s:string" />
161 </wsdl:message>
162 <wsdl:message name="GetLocationByIP_V2HttpPostOut">
163   <wsdl:part name="Body" element="tns:IP" />
164 </wsdl:message>
  
```

Types

```

82 <complexType name="IP">
83   <sequence>
84     <element minOccurs="0" maxOccurs="1" name="Error" type="tns:Err" />
85     <element minOccurs="0" maxOccurs="1" name="City" type="s:string" />
86     <element minOccurs="0" maxOccurs="1" name="Region" type="s:string" />
87     <element minOccurs="0" maxOccurs="1" name="Country" type="s:string" />
88     <element minOccurs="0" maxOccurs="1" name="CountryISO3" type="s:string" />
89     <element minOccurs="0" maxOccurs="1" name="CountryISO2" type="s:string" />
90     <element minOccurs="1" maxOccurs="1" name="Longitude" type="s:double" />
91     <element minOccurs="1" maxOccurs="1" name="Latitude" type="s:double" />
92     <element minOccurs="1" maxOccurs="1" name="Certainty" type="s:int" />
93     <element minOccurs="0" maxOccurs="1" name="IsProxy" type="s:string" />
94     <element minOccurs="0" maxOccurs="1" name="ProxyType" type="s:string" />
95     <element minOccurs="0" maxOccurs="1" name="ISP" type="s:string" />
96     <element minOccurs="0" maxOccurs="1" name="NetblockOwner" type="s:string" />
97     <element minOccurs="0" maxOccurs="1" name="Debug" type="s:string" />
98   </sequence>
99 </complexType>
  
```

What does response look like?

- Result of call to Web Service for an IP of 216.150.141.228

```

- <IP>
  <City>Tarrytown</City>
  <Region>NY</Region>
  <Country>United States</Country>
  <CountryISO3>USA</CountryISO3>
  <CountryISO2>US</CountryISO2>
  <Longitude>-73.839699</Longitude>
  <Latitude>41.083698</Latitude>
  <Certainty>95</Certainty>
  <IsProxy>FALSE</IsProxy>
  <ProxyType>NONE</ProxyType>
  <ISP>Xand Corporation</ISP>
  <NetblockOwner>XAND CORP</NetblockOwner>
  <Debug>
</IP>
  
```

Call from Straight Code



- From w/in a PHP Page we need a way to request a URL and get a result back
- Several options available
 - Code it yourself – it's "just" sockets
 - Use fopen
 - Use file_get_contents()
 - Use CURL

What to do with XML we get?



- Pull the "important" stuff out of xml, or parse it
- Two parsers available, SAX and DOM
- DOM makes sense on "smaller" XML docs with a consistent structure
- SAX makes sense on "larger" docs or inconsistent structures
- Small and large are relative, but DOM holds entire document in memory, SAX only holds one part at a time
 - 1K XML -> DOM; 1G XML -> SAX ; 10M XML -> ??
- IP Info is clearly DOM land
 - Small and well defined

PHP



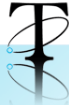
- PHP has native support for web services
- PHP is a dynamic language!
- PHP is (relatively) slow since it's interpreted at runtime
 - Time is relative. It's still blindingly fast!
- At runtime PHP parses WSDL and dynamically creates proxy methods
- WSDL is now runtime artifact
- Endpoints in WSDL must be correct
 - PHP uses them directly
 - Different WSDL needed for test / qa / production

PHP



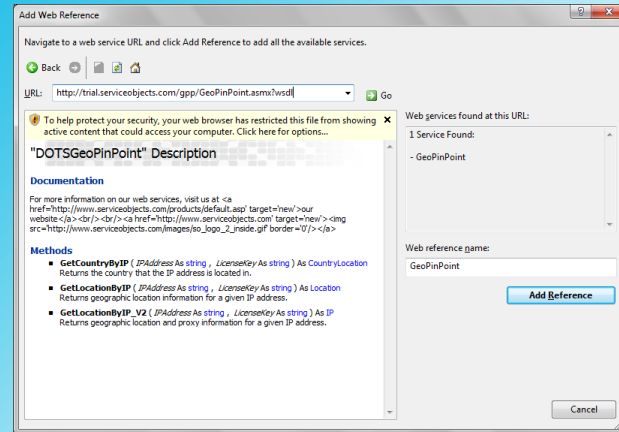
```
1 //Define soap client -- NB: The url is to the WSDL
2 $svc = new SoapClient('http://url-to-wsdl');
3
4 //Call service
5 $ipInfo = $svc->GetLocationByIP_V2( '216.150.141.228', 'LicenseKey');
6
7 //Load answer into local variable
8 $country = $ipInfo.Country;
9
10 //... or pass them to a function call
11 $myResult = MyFunction($ipInfo.Longitude, $ipInfo.Latitude);
12
```

.NET

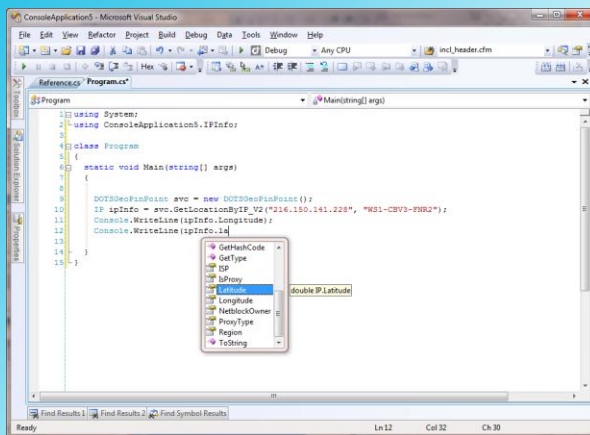


- Hybrid of PHP and RPG
- WSDL is parsed at development time to generate objects (data structures)
 - Speed of compiled objects
 - Same fragility problem as RPG, if the service changes...
- Parsing done automatically by wizard
- URL in WSDL doesn't have to be correct
 - You can specify at runtime; great for test/qa/production
- Visual Studio Wizard
 - Point it at the WSDL
 - Generates needed code and proxy method
- Call as if it was any other method

Visual Studio Wizard



Get IP Info

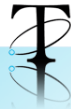


Web Service Providers



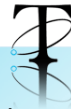
- Let's Pretend...
 - We sell "things"
 - We want to enable customers to check pricing
 - We have a pricing module already
- We need to define our interface
 - What are we doing about security?
 - What parms do we need?
 - How will consumers access our service? URL? SMTP?
 - Are we doing REST or SOAP?

Security!?!



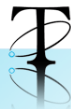
- Typically not an issue when consuming
 - you do what the provider tells you to
- As a provider
 - you have data that shouldn't be publicly available
 - DOS attacks
 - Data leakage
 - Competitors can walk your database
 - Fulfilling a requests costs you
 - Time
 - Bandwidth
 - Real \$ (Does service do something real?)
- SSL
 - Do we encrypt the tunnel? And where? SSL Accelerator?

Security Options



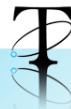
- Security by obscurity
 - No one knows about my service, so it's secure! Ha!
- Network security
 - This service only used internally
- Shared secret
 - "secret" that must be included on all calls
- HTTP 403 Username / Password
 - Uses normal HTTP authentication mechanism
- Session based
 - There's a login "service" that starts a session. Then session id is passed to each service as a parm
- Kerberos / SSO / Other advanced topics

What Parameters do we need?



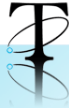
- Service Key (Shared Secret) from security decision
- Pricing is simple in this case, we do quantity breaks
- What are our parms to the existing pricing routine?
 - Item#, Quantity
- Our answer! We need:
 - Access Key
 - Item#
 - Quantity

How to access?



- URL? SMTP?
 - Yes, I said SMTP. SOAP does it all, if we want
- We want something simple
- We already have a deployed website we can integrate with
- Services are often segregated into a directory
 - Robots.txt file tells search engines to stay out
 - Makes it clear these are services
- URL should describe Service
- <http://mycompany.com/services/GetPricing.xxx>
 - xxx depends on the implementation, cgi, php, aspx, jsp, etc.

SOAP or REST



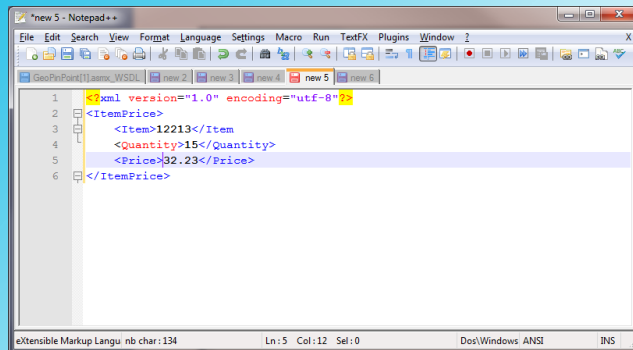
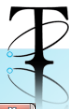
- How will our service be called?
 - Not sure? Customers do crazy things
 - Likely to be on a web page, let's make it simple
 - This is for external consumption
 - Not part of an internal SOA project
 - No need for orchestration or composition
 - Customers have expressed an interest in Ajax
- REST is a good fit here
 - Simple to consume, simple to expose!
 - Browser consumption almost always demands REST

What's our API?



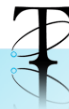
- Remember, REST is just HTTP!
- We're just retrieving, or GETting a price
 - the GET verb is all we need.
- Need to define some parameters on the URL
 - URL Parameters are specified as Key=Value
 - AccessKey, ItemNo, Qty
- `http://mycompany.com/services/GetPricing.php?AccessKey=f3s21&ItemNo=12213&Qty=15`
- That's how to call, what do we return?

Could simply be XML

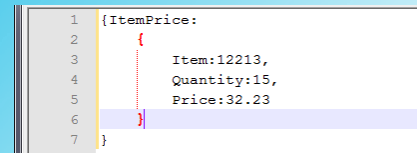


```
<?xml version="1.0" encoding="utf-8"?>
<ItemPrice>
  <Item>12213</Item>
  <Quantity>15</Quantity>
  <Price>32.23</Price>
</ItemPrice>
```

But we want simplicity!

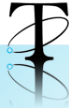


- Could use just plain text
 - Return "32.23" as sole response
- Use JSON – Javascript Object Notation
- Single Javascript function to parse this on browser side



```
{ItemPrice:
  {
    Item:12213,
    Quantity:15,
    Price:32.23
  }
}
```

Service is simply a web page!



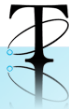
- Use your favorite web page development environment
 - CGI-DEV2, Java, PHP, .Net, etc.
- Instead of returning HTML you're returning JSON formatted text.
- Congratulations, you've written a web service
 - See, it's simple!

PHP SOAP Service?



- Exercise left to the reader
- Hint: SoapServer class
 - Specifically: SoapServer::addFunction

Questions?



Questions?

How to contact me:
Walden Leverich
WaldenL@techsoftinc.com
Twitter: [@WaldenL](https://twitter.com/WaldenL)
<http://blog.waldenl.com>