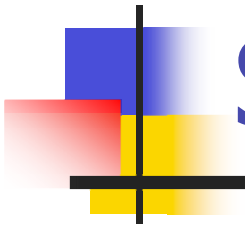
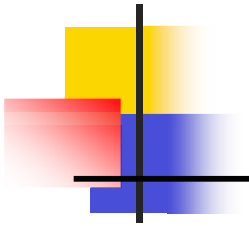


Interoperable Web Services: A Primer

Noel J. Bergman

DevTech®

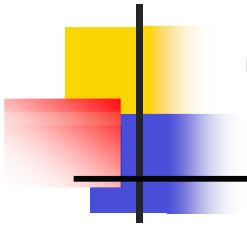


Session Overview

The Web Services specification arena has exploded. Some specifications are W3C, some are OASIS. Some are real, some are important, some are vendor hype, many are incompatible. No wonder people tend to refer to WS-* as WS-Deathstar!

Which one(s) do you really need to know and use? How can your Web services interoperate? In this session, we will explore those issues, with attention paid to development of interoperable Web services for .NET and Java. Perhaps even a surprise language or two.

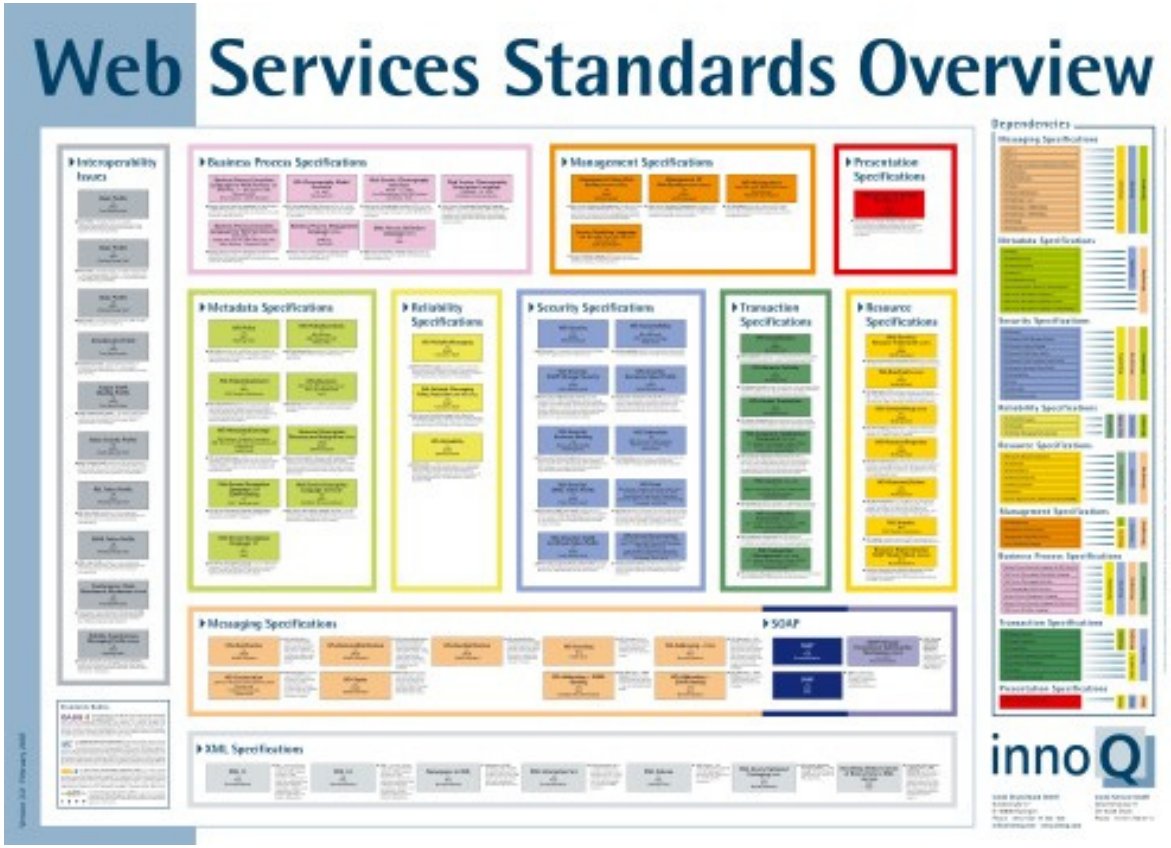
PLEASE ASK QUESTIONS! ☺



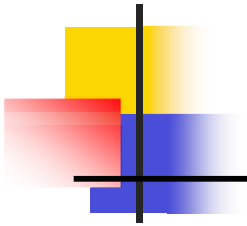
The Problem

http://www.josemercader.es/wp-content//mulita.jpg

The WS-* Landscape

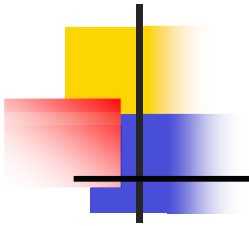


http://www.innoq.com/soa/ws-standards/poster/



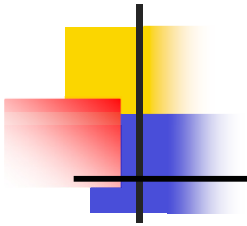
What's Going On?

- "WS was the reaction of the CORBA crowd to the web: complex and over-engineered."
  - Simon Phipps
- "Nobody cared about the slaves who died laboring to raise the Tower of Babel."
  - Fritz Lang, *Metropolis*



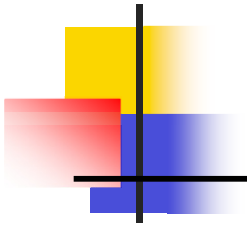
Sanjiva Speaks

- *"The biggest problem I see with the WS-* specs is the lack of a single spec that lays out the key components of the platform architecture. This results in people saying "geez there are 100+ WS-* specs; how can I possibly know what's important and what's not?!". Microsoft's WS-* platform, WCF, will help address this somewhat as that platform has a very small and clear set of specs but it'll take time to kill off all the political plays by various companies to create competing standards."*
  - Sanjiva Weerawarana (Co-author: WSDL, BPEL4WS, WS-Addressing, WS-RF, WS-Eventing, Apache SOAP, Apache Axis, …)
    - See also: http://benjaminm.net/PermaLink.aspx?guid=184a8f36-1639-4510-ac81-c6a8730f19c1
- *"As the person who did the first ever SOAP implementation (Apache SOAP) I take full responsibility for taking several wrong decisions on how WS-* should be implemented."* - Sanjiva
- Basically, Sanjiva, one of IBM's lead WS-* architects before he left to co-found WSO2 with Davanum "Dims" Srinivas and Paul Fremantle, is saying that out of the myriad WS-* specifications that are foisted on the marketplace by vendors, only a few matter.
  - So where can we get this list? Is Microsoft really canonical?



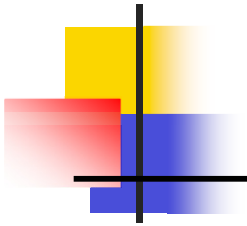
Not So Fast …

- *"A lot of the criticism of WS-* comes from people who continue to look at SOAP as a distributed object communication mechanism. SOAP 1.1 certainly had those traits (with SOAP-RPC and SOAP Encoding) but SOAP 1.2 has lost those totally and same for WSDL- 1.1 had RPCness built in but with 2.0 its not in the least."*
  - Sanjiva
- But for reasons we'll discuss shortly, SOAP 1.2, WSDL 2.0, and others are for the future, but not the present.
- So while it might be nice to hear that things are happening that might make WS-* easier in the future, what about today?



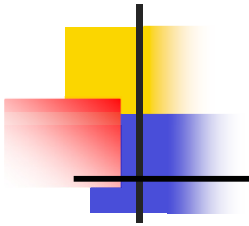
Emphasize Interoperability

- The whole point of using WS-* is interoperability. Else, why bother? We already have plenty of distributed computing solutions, and XML over the wire isn't the fastest and most compact solution.
- As noted by Simon Phipps, recognize that many vendors in the WS-* space are trying to recreate everything from existing local and distributed paradigms in WS-* terms. This is a major error, and root cause of the problem. Reject this, and "just say no!"
- Enter, the WS-I.



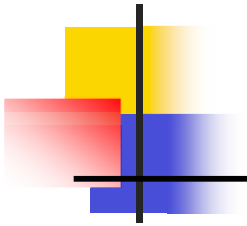
WS-I.ORG

- Web Services Interoperability Organization
- 75+ Members and 98+ "Advocates"
- Does not create WS-* specifications. Instead, the WS-I provides:
  - Profiles
  - Testing Tools
    - http://www.ws-i.org/deliverables/workinggroup.aspx?wg=testingtools
    - http://www.ibm.com/developerworks/webservices/library/ws-wsitest/
    - See also: http://www.soapui.org/userguide/tools/wsi.html
  - Sample Applications
    - http://www.ws-i.org/deliverables/workinggroup.aspx?wg=sampleapps
- The WS-I Profiles are your key to WS-* Interoperability.



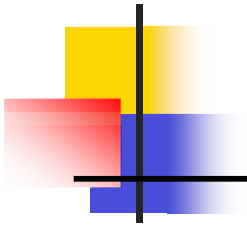
WS-I Profiles

- WS-I Profiles define how to use existing WS-* in order to achieve interoperability. We don't use everything that the specifications allow. We restrict ourselves to an interoperable subset as prescribed by a WS-I profile.
- Basic Profile
  - Specifies how to use WSDL, SOAP, HTTP, XML Schema.
- Basic Security Profile
  - Specifies how to use WS-Security



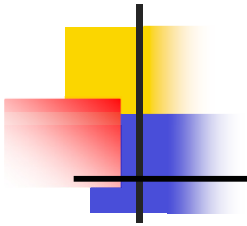
Basic Profile

- How to use WSDL, SOAP, HTTP, XML Schema to ensure interoperability.
- Basic Profile 1.0
  - SOAP 1.1, WSDL 1.1, XML Schema, HTTP 1.1
    - WSDL 1.1: http://www.w3.org/TR/2001/NOTE-wsdl-20010315
    - SOAP 1.1: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
    - XML Schema Part 1 – Structures: http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/
    - XML Schema Part 2 – Datatypes: http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/
- Basic Profile 1.1
  - As above, but SOAP binding is factored into a companion specification, and SOAP attachments (SwA) are in a second companion specification.
  - SwA is not universally implemented (*q.v.*, MTOM).



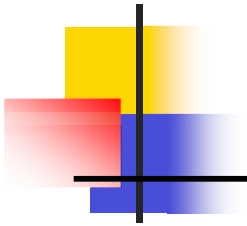
Basic Profile

- Basic Profile 1.2 (**DRAFT**)
  - http://www.ws-i.org/Profiles/BasicProfile-1.2.html
  - Adds support WS-Addressing and MTOM, and brings SOAP binding back into the core.
    - WS-Addressing: http://www.w3.org/Submission/ws-addressing/
    - WS-Addressing SOAP Binding: http://www.w3.org/TR/2005/CR-ws-addr-soap-20050817/
- Basic Profile 2.0 (**FUTURE**)
  - Will add support for SOAP 1.2 and XOP. Nota bene: the Working Group explicitly states that it will **not** add support for WSDL 2.0.



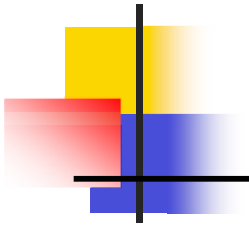
Basic Security Profile

- How to use WS-Security
  - www.oasis-open.org/committees/wss/
  - http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
  - http://www.oasis-open.org/committees/download.php/16792/oasis-200512-wss-soap-message-security-1.0-errata-005.pdf
  - These are for WS-Security 1.0. Basic Security Profile 1.1 is currently a draft, and will add support for WS-Security 1.1.
- Companion Specifications for Security Tokens:
  - Rights Expression Language (REL)
  - Kerberos
  - SAML
  - UserName
  - X.509
- A nice thing about WS-Security is that you are not likely to touch it as a programmer. It is generally handled by configuring your WS runtime, not by coding within your Web service. Unfortunately, although the content sent between the WS stacks is covered by the specification, configuring each WS stack is a vendor-specific process.



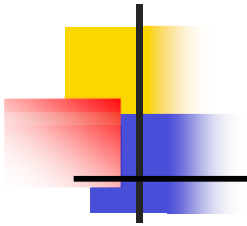
Reliable Secure Profile

- Currently under development.
- A draft of usage scenarios is available.
  - http://www.ws-i.org/profiles/rsp-scenarios-1.0.pdf
- Will Cover:
  - WS-I Basic Profile 1.2 and 2.0
  - WS-I Basic Security Profile 1.0 and 1.1
  - WS-ReliableMessaging 1.1
    - Co-chaired by Paul Fremantle
    - http://docs.oasis-open.org/ws-rx/wsrm/v1.1/wsrm.html
  - WS-SecureConversation
    - Co-chaired by Kelvin Lawrence
    - http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.3/ws-secureconversation.html



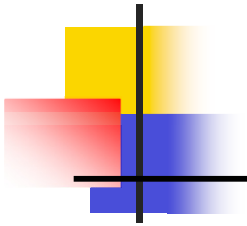
Summary So Far

- Basic Profile 1.0 and 1.1 are the most widely implemented. They cover:
  - SOAP 1.1, WSDL 1.1, XML Schema, HTTP 1.1
  - Basic Profile 1.1 plus its companion Simple SOAP Binding Profile is the rough equivalent to Basic Profile 1.0 with errata. The companion Attachments Profile is not universally implemented, and therefore cannot truly be considered interoperable.
  - WS-Addressing, MTOM, *etc.* are not covered by any non-draft version of the WS-I Basic Profile.
- Basic Security Profile 1.0 adds WS-Security, and is somewhat widely supported for UserName and X.509 Security Tokens.
- Therefore, Basic Profile 1.0 with errata (or Basic Profile 1.1 plus Simple SOAP Binding Profile) and optionally Basic Security Profile 1.0 are what you should work with **today** if you want interoperability. You want to use a WS toolset that will check for and facilitate adherence to WS-I Profiles.
- On-going Profile development efforts at the WS-I add only WS-Addressing, MTOM, XOP, WS-ReliableMessaging and WS-SecureConversation to the set. No other WS-* specifications are covered by the organization *"chartered to promote Web services interoperability across platforms, operating systems and programming languages."*



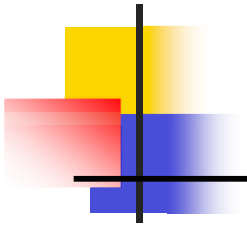
Next Steps

- WSDL Basics
- XML Schema Basics
- Java <-> WSDL Mapping
  - JAX-RPC, JAX-WS (w/ JAXB)
  - WSEE (JSR-109, JSR-921)
- Interoperability Tips
- Demos



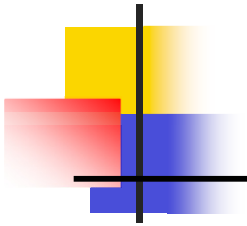
WSDL Basics

- Web Service Definition Language
  - One definition of Web service, albeit not a great one, is that if we can define it in WSDL, it is a Web service.
- In WSDL 1.1, Web services are defined by:
  - *Types*
  - *Messages*, which use the types
  - *PortTypes*
    - *Operations*, which use the messages
  - *Bindings*
  - *Services*
- A *PortType* is the abstract interface to the service.
- A *Binding* tells us how to represent that PortType, *e.g.*, SOAP over HTTP or SOAP over JMS.
- A *Service* tells us where to find the service.
- We'll go into these in more detail.



WSDL Essential Outline

```
<?xml version="1.0"?>
<definitions targetNamespace="…"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types> XML Schema defined types </types>
  <message name="…"> message parts using defined types </message>
  <portType name="…"> operations generically defined by messages </portType>
  <binding name="…" type="…"> how to represent the portType on the wire </binding>
  <service name="…">
    <port name="…" binding="…">
      <soap:address location="URI"/>
    </port>
  </service>
</definitions>
```

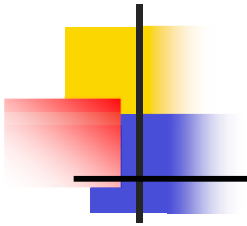


WSDL Types

- We use XML Schema to define the structure of data that we exchange with messages between our services.

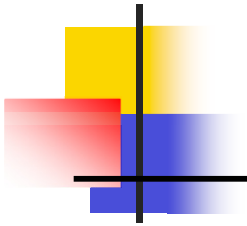
```
<?xml version="1.0"?>
<definitions targetNamespace="…"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
             xmlns="http://schemas.xmlsoap.org/wsdl/">
  …
  <types>
    <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="…">
      … the types we define for use in our interface …
    </xsd:schema>
  </types>
  …
</definitions>
```

- We can also use `xsd:import` to import a schema file, or `wsdl:import` to import a WSDL document containing the `wsdl:types` section.



XML Schema

- The WSDL Types are defined using XML Schema.
- Let's look at a few examples.
- For more examples and an entire session on XML Schema, see Neil Graham's 2005 presentation, *Who's Afraid of XML Schema?*
  - http://www.softwaresummit.com/2005/speakers/GrahamSchema.pdf

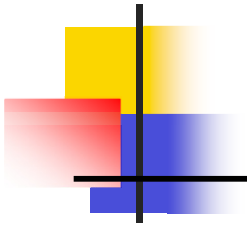


WSDL Messages

- Having defined types, we now define messages.
- A message has one or more parts, each a type or element previously defined.

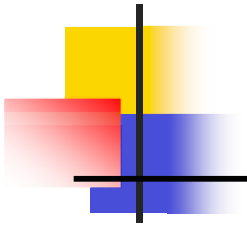
```
<message name="BinaryIntOp">
   <part name="operand1" type="xsd:int"/>
   <part name="operand2" type="xsd:int"/>
</message>

<message name="IntOpResponse">
   <part name="result" type="xsd:int"/>
</message>

<message name="CreateOrderRequest">
   <part name="order" element="my:CreateOrder"/>
</message>
```

- Next, we will create operations based on our messages.



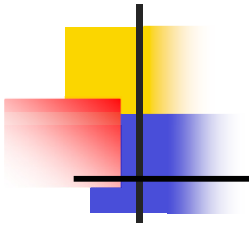
Defining the Interface

- In WSDL 1.x, we call the interface to our Web service a `portType` (in WSDL 2.0 sanity returns, and the element is named `interface`).
- A `portType` acts as a container for a set of `operation` elements, defining the abstract interface to our Web service.



WSDL Operations

- WSDL Operations are made up of messages. There are three messaging elements that can be associated with an operation.
  - `<input>`
  - `<output>`
  - `<fault>`
- The order of the `<input>` and `<output>` elements matters, and defines one of four (in WSDL 1.x) message exchange patterns:
  - Request-response: `<input>` followed by `<output>`
  - One-way: `<input>`
  - Notification: `<output>`
  - Solicit-response: `<output>` followed by `<input>`
- The WS-I mandates against the latter two MEPs.
- The `<fault>` element is used as the entire response to indicate a failure (*e.g.*, client side processing, server side processing, or service exception).



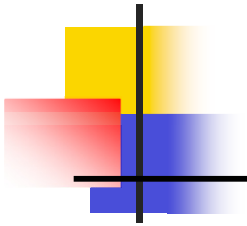
Sample Interfaces

- A calculator with two math operations:

```
<portType name="Calc">
     <operation name="AddInt">
          <input message="BinaryIntOp"/>
          <output message="IntOpResponse"/>
     </operation>
     <operation name="SubtractInt">
          <input message="BinaryIntOp">
          <output message="IntOpResponse"/>
     </operation>
</portType>
```

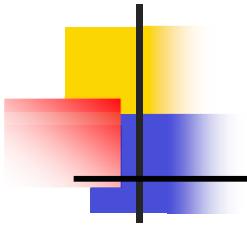
- An order system with one operation:

```
<portType name="OrderSystem">
     <operation name="CreateOrder">
          <input message="CreateOrderRequest"/>
          <output message="CreateOrderResponse"/>
     </operation>
</portType>
```



WSDL Bindings

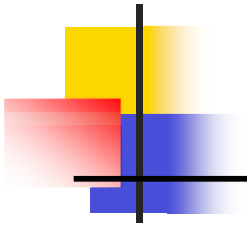
- So far, we have defined an abstract interface whose operations are built from combinations of messages containing data of defined types.
- But we have not yet defined how to represent and transport this information on the wire. This is the role of a *binding*.
  - http://www.w3.org/TR/2001/NOTE-wsdl-20010315#_soap-b
- WS-I mandates that we define a SOAP binding over HTTP.
  - WS-I also mandates the use of HTTP POST, and 500 status response for SOAP faults.
- We could optionally define additional bindings, *e.g.*, SOAP over JMS or SOAP over AMQP, although such are not actually permitted by WS-I.



Calculator Binding

- Sample binding for the calculator service:

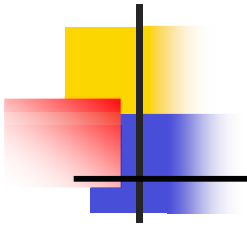
```
<wsdl:binding name="CalcSoapBinding" type="Calc">
     <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http" />
     <wsdl:operation name="AddInt">
            <soap:operation soapAction=""/>
            <wsdl:input><soap:body use="literal"/></wsdl:input>
            <wsdl:output><soap:body use="literal"/></wsdl:output>
     </wsdl:operation>
     <wsdl:operation name="SubtractInt">
            <soap:operation soapAction=""/>
            <wsdl:input><soap:body use="literal"/></wsdl:input>
            <wsdl:output><soap:body use="literal"/></wsdl:output>
     </wsdl:operation>
</wsdl:binding>
```



Order System Binding

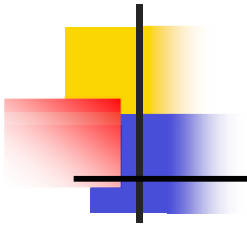
- Sample binding for the Order System service:

```
<wsdl:binding name="OrderSystemSoapBinding" type="OrderSystem">
    <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="CreateOrder">
        <soap:operation soapAction=""/>
        <wsdl:input>
            <soap:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
```



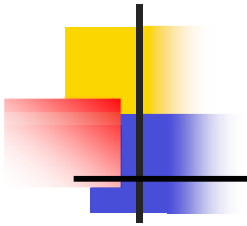
Message Encoding Styles

- Specifies how to format the message data.
- Literal
  - The message is formatted according to the XML schema.
- Encoded
  - Banned by WS-I, so for our purposes, Section 5 of the SOAP specification does not exist.
  - Microsoft's *The Argument Against SOAP Encoding*:
    - http://msdn2.microsoft.com/en-us/library/ms995710.aspx



Messaging Binding Styles

- RPC
  - When the request message is encoded, it will be automatically wrapped in an XML element whose name is the name of the operation being requested.
  - .NET does not support RPC style, deeming it unnecessary:
    - http://msdn2.microsoft.com/en-us/library/ms996466.aspx
- Document
  - The message is treated as a document, with no wrapper created.
  - Document Literal Wrapped
    - Although this is often referred to conversationally as a third "style", it is not a true style; it is a convention we often use with Document style. We create our own wrapper element in the XML schema, where the name of the element is the name of the operation with which it will be uniquely used. Thus the wire level representation of RPC/literal and Document/literal is the same when the later uses a wrapper element.
    - http://atmanes.blogspot.com/2005/03/wrapped-documentliteral-convention.html
- *Nota Bene*: message binding style does **not** equal programming model, as evidenced by Document Literal Wrapped.

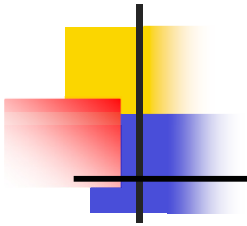


WSDL Service and Port

- The WSDL Port specifies an end point with which to communicate.

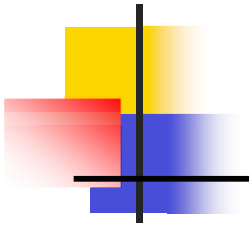
```
<service name="CalcServiceService">
    <port name="CalcService" binding="CalcSoapBinding">
        <soap:address location="http://..."/>
    </port>
</service>
```

- There are other means by which to specify the end point address, so this may be considered an optional part of a WSDL document. It may also be generated by a service registry.



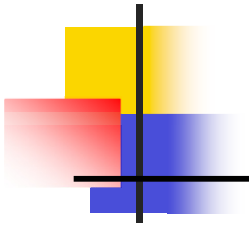
Finishing the WSDL

- Now we have both defined an abstract interface, the `portType`, and defined a mapping to a SOAP format over a wire level protocol.
- The last thing that we need to do is to indicate the end point to which we will communicate.
- WS-Addressing will change how we do this, but for now …



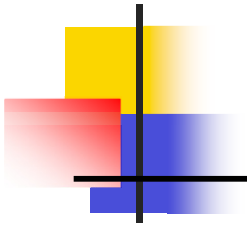
Coding Your Web Service

- Java
  - JAX-RPC provides Java support for Web service clients and providers.
    - JAX-WS is the next generation.
    - For historical reasons, JAX-RPC includes its own support for XML <-> Java marshalling. JAX-WS defers to JAXB, as do other specifications.
  - We can generate WSDL from Java, and we can generate Java both client and server side code from WSDL.
  - JSR-109 (EWS 1.0) / JSR-921 (EWS 1.1) specify how Web services fit into the Java EE Web and EJB container environment.
    - One rather ugly bit is that the `<servlet-class>` element in `web.xml` is overloaded to refer to a POJO Web service, which requires a suitable Web container this is compliant with these additional requirements.
- Microsoft .NET provides similar support.
- Python
  - http://www.diveintopython.org/soap_web_services/
  - http://pywebsvcs.sourceforge.net/



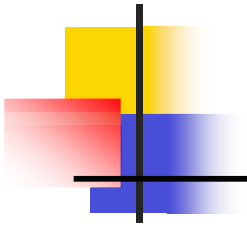
Interoperability Tips

- Web services should be stateless.
  - Each request should carry all of the necessary context, and not expect that the service remembers context from one request to another.
  - Consider context in the response, too. Although response context may be implicit in the RPC programming model, other programming models may require context to be carried in the response.
- Starting from WSDL will prevent you from doing things in Java that can be expressed in WSDL, but may not be WS-I compliant or compatible with a non-Java WS stack (*e.g.*, .NET)
- Avoid using Java (or other language) Collection classes in the interface to your service. Stick with XML schema defined structures and with plain XML sequences defined using minoccurs/maxoccurs.
- Avoid arrays containing null elements.
- When starting from the language side, rather than starting from WSDL, do not use overloaded methods.



Related Sessions

- WS-Security
  - *Security 101: Security Core Concepts... Where Do You Start?* – Denise Hatzidakis
  - *Web Service Security by Example* – Denise Hatzidakis
- WS Message Transfer Alternatives to HTTP
  - *Building a Reliable Messaging Infrastructure Using Apache ActiveMQ* – Bruce Synder
  - *Building Scalable Enterprise Messaging Systems Using AMQP* – Rajith Attapattu
- Service Oriented Architecture (SOA)
  - *Building a Service Oriented Architecture Using the Apache ServiceMix ESB* – Bruce Synder
  - *The Zen or Tao of SOA and SaaS: What's Real, What's Hype* – David Moskowitz
- REST
  - *WS-*? Just give it a REST!* – Noel J. Bergman
  - *REST in Peace* – Rajith Attapattu
- AXIS
  - *Quickstart Axis2/Java: From Newbie to SOAP Guru* – Deepal Jayasinghe
  - *Real World Axis2/Java: Highlighting Performance and Scalability* – Deepal Jayasinghe



Demonstration

Let's fire up an IDE, and inspect some live and interoperable Web services.

(and yes, I *could* do this in emacs ☺)

Thanks for coming. ☺

Please turn in the session evaluation.