Service Oriented Architecture, Web Services and Web 2.0: Applications, Infrastructure, Standards

Section 2

Dr. Donald F. Ferguson
donald.ferguson@ca.com

SVP, Distinguished Engineer
Chief Architect for Enterprise IT Management
Agenda

• My philosophy for this class 11:00

• Focus Questions’ Answers 11:10
  – Building a basic Web service for Person
  – Top-Down versus Bottom-up.
  – Email binding for CRUD for person.
  – What did we learn?

• Break 11:50

• The 1st Look Under the Hood 12:00
  – Introducing the Simple Object Access Protocol (SOAP) 12:00
  – Introducing WSDL 12:15
  – Container 101 12:30

• Discuss 2nd Assignment 12:40
Agenda

• **My philosophy for this class**  11:00

• **Focus Questions’ Answers**  11:10
  – Building a basic Web service for Person
  – Top-Down versus Bottom-up.
  – Email binding for CRUD for person.
  – What did we learn?

• **Break**  11:50

• **The 1\textsuperscript{st} Look Under the Hood**  12:00
  – Introducing the Simple Object Access Protocol (SOAP)  12:00
  – Introducing WSDL  12:15
  – Container 101  12:30

• **Discuss 2\textsuperscript{nd} Assignment**  12:40
My Philosophy

• Our relationship is complex. You are my employer
  – Medieval universities worked this way.
    • Faculty were paid by each student in the class.
    • Students could fine professors for a bad lecture.
  – But, I cannot make the class too easy and give everyone an A.
    • This would not be in your best interests.
    • You are paying me to teach you things that prepare you for life, and more specifically your career.

• What is the software development approach in industry?
  – Fundamentally, there are three core roles.
    • Managers
    • Developers, who also design and architect.
    • Technical leads (architects)
  – The same persona may play more than one role.
My Philosophy

• In this class,
  – I am playing the role of technical lead.
  – You are playing the role of developer.

• Technical leads
  – Help the developers complete their tasks.
    • Assign tasks to developers.
    • Provide guidance.
    • Review work.
    • **Answers questions and help a developer who is having trouble.**
      – **And teach the developer how to do their job.**

• This means
  – Asking for help and clarification is OK.
  – Getting the wrong answer or design is OK.
  – If you ask for help and
    • I do not answer your email or respond
    • You cannot complete your task after my help.
    • I am the one who made a mistake not you.
  – **Not completing your task and not asking for help is not OK.**
My Philosophy

• I am an atypical team lead, however.
  – I know that people will not ask for help and fail.
  – There are many reasons: pride, shyness, time pressure, personal issues.
  – Not asking for help shows initiative.
  – I cannot help you until you are ready.

• How I work
  – I will help you when you ask for it.
  – If you do not complete an assignment,
    • I will give advice.
    • You can try again.
    • You can fail again.
  – Your grade will be based on the quality of your work.
    • Relative to other developers.
    • You can always improve a previous assignment if you want.

• Reference information
  – Books are rarely useful, or only help with certain aspects of a project.
  – In your career, you will need to be able to
    • Find information yourself.
    • Filter and process the information.
    • Apply the information.
  – Your team leads cannot and will not provide too many pointers.
    • They are guiding 12 people on 12 tasks.
    • The information changes to fast.
Agenda

• My philosophy for this class  11:00
• Focus Questions’ Answers  11:10
  – Building a basic Web service for Person
  – Top-Down versus Bottom-up.
  – Email binding for CRUD for person.
  – What did we learn?

• Break  11:50
• The 1st Look Under the Hood  12:00
  – Introducing the Simple Object Access Protocol (SOAP)  12:00
  – Introducing WSDL  12:15
  – Container 101  12:30
• Discuss 2nd Assignment  12:40
Building a Basic Web Service: Your 1\textsuperscript{st} Assignment

- **WSO2** – www.wso2.com
  - Download and install application server.  
  - Read the tutorials for building a simple Web service.
    - WSO2 has tutorials
    - http://ws.apache.org/axis2/1_4_1/contents.html
    - Many Axis2 Tomcat documents on the Web

- **Your 1\textsuperscript{st} Project**
  - Define a simple person object, e.g. name, id, etc.
  - Implement a POJO Web service for CRUDing a person.
  - Write a simple Java client for calling the service.
  - Explain the concepts and technology that you used and their relationship (5-6 pages).

I explained and used this term in class but should have not used the acronym in the assignment write up without first defining it.

What should you have done?
Solving Assignment 1, Part 1
A Simple POJO Web Service

• Approach
  – Use Web search
  – Find a tutorial, all of which come with sample code.
  – Follow the tutorial and modify the code.
  – Some examples that I found in less than 30 minutes
    • [http://ws.apache.org/axis2/1_4_1/pojoguide.html](http://ws.apache.org/axis2/1_4_1/pojoguide.html)

• How do you complete the project?
  – Define and implement the Java interface and class.
  – Unit test the class from a Java main().
  – Run produce the WSDL, or use a generator.
  – Follow the 3 steps to deploying the service.
  – Use one of the default browser based Web service testers.
  – Modify your Java main to use the generated stub.
Solving Assignment 1, Part 1
A Simple POJO Web Service

• Approach
  – Use Web search
  – Find a tutorial, all of which come with sample code.
  – Follow the tutorial and modify the code.
  – Some examples that I found in less than 30 minutes
    • http://ws.apache.org/axis2/1_4_1/pojoguide.html
    • http://www.eclipse.org/webtools/jst/components/ws/1.5/tutorials/BottomUpWebService/BottomUpWebService.html
    • http://www.devx.com/Java/Article/33839

• How do you complete the project?
  – Define and implement the Java interface and class.
  – Unit test the class from a Java main().
  – Run produce the WSDL, or use a generator.
  – Follow the 3 steps to deploying the service.
  – Use one of the default browser based Web service testers.
  – Modify your Java main to use the generated stub.
Solving Assignment 1, Part 1

An example of what you could have followed.

http://www.devx.com/Java/Article/33839

(Switch to browser windows)
Focus Questions

• Compare and contrast “top-down” versus “bottom up” development of POJO Web services.
  – What are top-down and bottom-up.
  – What are the pros and cons

• Propose and define an email binding
  – How would you add support for invoking the service and receiving the response by sending email?
  – Explain the design.
  – What are some of the complexities?

• 2-3 pages for each focus question.
Focus Questions

• Compare and contrast “top-down” versus “bottom up” development of POJO Web services.
  – What are top-down and bottom-up.
  – What are the pros and cons

• Propose and define an email binding
  – How would you add support for invoking the service and receiving the response by sending email?
  – Explain the design.
  – What are some of the complexities?

• 2-3 pages for each focus question.
Your development project would have started with one of two different artifact.

This

```java
public interface AccountManager {
    public Account getAccount(String accountId);
    public void setStatus(String accountId, Boolean status);
}
```

Or
  <wsdl:element name="getAccount">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="accountId" nillable="true" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </wsdl:element>

  <wsdl:element name="getAccountResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="return" nillable="true" type="ns0:Account" />
      </xs:sequence>
    </xs:complexType>
  </wsdl:element>

  <wsdl:element name="setStatus">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="accountId" nillable="true" type="xs:string" />
        <xs:element name="status" nillable="true" type="xs:boolean" />
      </xs:sequence>
    </xs:complexType>
  </wsdl:element>

    <xs:element name="Account" type="ax23:Account" />
    <xs:complexType name="Account">
      <xs:sequence>
        <xs:element name="accountId" nillable="true" type="xs:string" />
        <xs:element name="address" nillable="true" type="xs:string" />
        <xs:element name="firstName" nillable="true" type="xs:string" />
        <xs:element name="lastName" nillable="true" type="xs:string" />
        <xs:element name="status" type="xs:boolean" />
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
</wsdl:definitions>
Top Down versus Bottom Up

• Top down
  – Cons
    • WSDL is large and complex.
    • Burdens a Java developer with a lot of complex new stuff.
    • Puts the complex stuff in the critical path of making progress.
    • Creates several files. In software development
      Many files + Large files + Cryptic files = Very bad.
  – Pros
    • Language neutral. What if the developer writing the calling application is using
      C# and Visual Studio, PHP or a workflow language, e.g. BPEL.
    • Easier approach to reference and reuse types and interfaces that are internet
      addressable.
      – There may be a standard out there or something you can reuse.
      – Your interfaces are Web documents, which makes some collaboration scenarios
        easier.
    • There are some capabilities that are hard to express in Java, e.g. Out-In
      Message Exchange Pattern

• Bottom up
  – Pros are top down’s cons
  – Cons are top down’s pros.
Focus Questions

• Compare and contrast “top-down” versus “bottom up” development of POJO Web services.
  – What are top-down and bottom-up.
  – What are the pros and cons

• Propose and define an email binding
  – How would you add support for invoking the service and receiving the response by sending email?
  – Explain the design.
  – What are some of the complexities?

• 2-3 pages for each focus question.
Solving Assignment 1, Part 2

• Think
  – Why did Don assign part 1?
  – Did part 1 teach me something I need to know for part 2?
  – Or is Don insane?

• The part 1 project produced
  – An implementation skeleton. Your code goes here.
  – A stub
  – A non-Java interface definition, basically spooky text.
  – An HTTP addressable place to call the service, and
  – A Web page for testing the service.

• What do I need to support email?
  – The implementation should stay the same.
  – The stub needs to send email, not SOAP. The unit test program is email.
  – How do I define the interface in email terms?
  – Where do I send/receive/respond to email.
  – How do I encode the operation (verb) and parameters?
A Solution

• Examining what the browser unit test does would have shown that invoking the service is something like http://www.donald-ferguson.net/PersonService?operation=create&lastname=Ferguson&... ...

• Which means I need an email template of the form

From: donff2@aol.com

To: personService@donald-ferguson.net

Subject: Operation = Request

Lastname = "Ferguson";
Firstname = "Donald";
IQ = "50";
Implement the Transport

Some mailbox somewhere.

- Use an API:
  - POP3
  - Outlook
  - Just script the gmail Web pages

The email netherworld.

Write a demon or something that:
- Reads email
- Parses the text
- Calls your Web service through the prebuilt client
- Forms the response
- Sends it.
Agenda

• My philosophy for this class 11:00
• Focus Questions’ Answers 11:10
  – Building a basic Web service for Person
  – Top-Down versus Bottom-up.
  – Email binding for CRUD for person.
  – What did we learn?
• Break 11:50
• The 1st Look Under the Hood 12:00
  – Introducing the Simple Object Access Protocol (SOAP) 12:00
  – Introducing WSDL 12:15
  – Container 101 12:30
• Discuss 2nd Assignment 12:40
What did we Learn?

• A service needs to
  – Describe its interface in one or more models (Java, WSDL, HTTP Get).
  – Encode (not cryptographically) the message.
  – Have an address that the caller can find.
  – Typically more than one binding.
    • HTTP/SOAP
    • Java Messaging Service
    • Email

• Should
  – Separate the implementation from the transport.
  – Completely encapsulate the implementation technology.
Agenda

• My philosophy for this class  11:00
• Focus Questions’ Answers  11:10
  – Building a basic Web service for Person
  – Top-Down versus Bottom-up.
  – Email binding for CRUD for person.
  – What did we learn?

• Break  11:50

• The 1st Look Under the Hood  12:00
  – Introducing the Simple Object Access Protocol (SOAP)  12:00
  – Introducing WSDL  12:15
  – Container 101  12:30

• Discuss 2nd Assignment  12:40
What did we learn about and what do we need??

• A service needs to
  – Describe its interface in one or more models (Java, WSDL, HTTP Get).
  – Encode (not cryptographically) the message.
  – Have an address that the caller can find.
  – Typically more than one binding.
    • HTTP/SOAP
    • Java Messaging Service
    • Email

• Web Services
  – Describes the interface in the Web Service Description Language.
  – Encodes the message in SOAP.
  – Has a default, standard binding to HTTP POST/GET.
SOAP Structure

Each SOAP message will have:

- An **Envelope**
  - How do you insert the data
  - In what the transport sends and receives.

- **A Header** (optional)

- **A Body**
  - What is the operation?
  - What is the data?

- The **Body** *may contain a Fault element.*
  - What was the exception?
Stock Quote example embedded in an HTTP POST Request

- POST /StockQuote HTTP/1.1
  Host: www.stocksserver.com
  Content-Type: text/xml; charset="utf-8" ← SOAP HTTP Header info
  Content-Length: nnnn
  SOAPAction: “Some-URI” ← SOAP HTTP Header info
  <env:Envelope
    xmlns:env="http://www.w3.org/2001/06/soap-envelope">  
    <env:Body>
      <m:GetStockQuote xmlns:m="Some-URI"
        env:encodingStyle="http://www.w3.org/2001/06/soap-encoding">
        <symbol>SUNW</symbol>
      </m:GetStockQuote>
    </env:Body>
  </env:Envelope>

- Sent from the client to the service. First few lines are HTTP headers and SOAP HTTP headers. Followed by the SOAPAction HTTP header and SOAP Envelope XML element. The SOAP Body contains application defined element(s), as defined by the service.
A Sample SOAP Message

```xml
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
     <getProductDetails
        xmlns="http://warehouse.example.com/ws">
       <productID>827635</productID>
     </getProductDetails>
   </soap:Body>
</soap:Envelope>
```
A Sample SOAP Message (cont)

The SOAP header provides a space for arbitrary headers to be added to the message.
A Sample SOAP Message (cont)

The contents of the SOAP body element can be any valid XML that the parties wish to interchange.
Using SOAP Headers

• Some simple examples
  – Add a signature to ensure the message isn’t modified
  – Add a process identifier to track this message as part of a wider process.
  – Add a userid so that end-to-end security can be guaranteed.
  – Add a message number so messages can be resent if lost.

• Many WS-* standards make sure the headers are interoperable.
  – WS-ReliableMessaging
  – WS-Security
  – WS-Coordination
The Role of the Container

Security Header
Reliable Messaging Header
Atomic Transaction Header

SOAP Message

Policy Declarations

This is fragile,
changes over time,
complex for business programmers,
error prone,
etc.

```java
double deposit(Message m) {
    checkForDuplicate(m.seqNo);
    registerForTransaction(m.context);
    isCAValid(m);
    checkSignature(m);
    updatePerformanceInfo();
    balance += m.amount;
    // ... ...
    updatePerformanceInfo();
}
```

© Donald F. Ferguson 2008
Container is a set of *policy driven* functions. Interceptor pattern for business logic and “stubs.” Before and After factoring of code.
Web Service Standards (Partial)
WSDL Document

• Types
  – Used to define custom message types

• Messages
  – Abstraction of request and response messages that my client and service need to communicate.

• PortTypes
  – Contains a set of operations.
  – Operations organize WSDL messages.
  – Operation-&gt;method name, PortType-&gt;java interface

• Bindings
  – Binds the PortType to a specific protocol (typically SOAP over http).
  – You can bind one PortType to several different protocols by using more than one port.

• Services
  – Gives you one or more URLs for the service.
  – Go here to execute “echo”.

© Donald F. Ferguson 2008
What Does echoServiceInterface Look Like In WSDL?

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://grids.ucsd.edu:8045/GCWS/services/Echo"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
    <wsdl:types />
    <wsdl:message name="echoResponse">
        <wsdl:part name="echoReturn" type="xsd:string" />
    </wsdl:message>
    <wsdl:message name="echoRequest">
        <wsdl:part name="in0" type="xsd:string" />
    </wsdl:message>
    <wsdl:portType name="Echo">
        <wsdl:operation name="echo" parameterOrder="in0">
            <wsdl:input message="impl:echoRequest" name="echoRequest" />  
            <wsdl:output message="impl:echoResponse" name="echoResponse" />
        </wsdl:operation>
    </wsdl:portType>
</wsdl:definitions>
```

© Donald F. Ferguson 2008
What Does This Look Like In WSDL, Continued?

```xml
<wsdl:binding name="EchoSoapBinding" type="impl:Echo">
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="echo">
    <wsdlsoap:operation soapAction="" />
    <wsdl:input name="echoRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                      namespace="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo"
                      use="encoded" />
    </wsdl:input>
    <wsdl:output name="echoResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                      namespace="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo"
                      use="encoded" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:service name="EchoService">
  <wsdl:port binding="impl:EchoSoapBinding" name="Echo">
    <wsdlsoap:address location="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo" />
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```
What Does echoServiceInterface Look Like In WSDL?

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://grids.u.cs.indiana.edu:8045/GCWS/services/Echo"
  xmlns="http://schemas.xmlsoap.org/wsdl/
  xmlns:apachesoap="http://xml.apache.org/xml-soap"
  xmlns:impl="http://grids.u.cs.indiana.edu:8045/GCWS/services/Echo"
  xmlns:intf="http://grids.u.cs.indiana.edu:8045/GCWS/services/Echo"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:types />
  <wsdl:message name="echoResponse">
    <wsdl:part name="echoReturn" type="xsd:string" />
  </wsdl:message>
  <wsdl:message name="echoRequest">
    <wsdl:part name="in0" type="xsd:string" />
  </wsdl:message>
  <wsdl:portType name="Echo">
    <wsdl:operation name="echo" parameterOrder="in0">
      <wsdl:input message="impl:echoRequest" name="echoRequest" />
      <wsdl:output message="impl:echoResponse" name="echoResponse" />
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```
Writing WSDL

• One could write WSDL by hand, but this is not the usual way.
• Most development environments will generate WSDL from annotations in a Java file, e.g.

```java
package helloservice.endpoint;

import javax.ws.rs.WebService;

@WebService()
public class Hello {
    private String message = new String("Hello, ");

    public void Hello() {}

    @WebMethod()
    public String sayHello(String name) {
        return message + name + ".";
    }
}
```
Graphical view of WSDL

- Or via a tool.
Agenda

• My philosophy for this class 11:00
• Focus Questions’ Answers 11:10
  – Building a basic Web service for Person
  – Top-Down versus Bottom-up.
  – Email binding for CRUD for person.
  – What did we learn?

• Break 11:50

• The 1st Look Under the Hood 12:00
  – Introducing the Simple Object Access Protocol (SOAP) 12:00
  – Introducing WSDL 12:15
  – Container 101 12:30

• Discuss 2nd Assignment 12:40
Assignment 2

• Development project
  – Expand and improve the Person project
    • Complete if you had trouble, did not finish or did not complete.
    • Use the JAX-WS style for defining the interface.
    • Show the separation between the skeleton and POJO, if you think that it is necessary.
  – Define an FTP and file system binding
    • Call the service by using FTP to put a comma separate file somewhere.
    • Call the service
    • Get the response
    • Put it in the file system somewhere.
  – Describe what you did.

• Focus Questions
  – Why are SOAP headers better than putting information in the body of the message?
  – Read about Apache Handlers
  – Write a short design of a logging service that extracts processes a SOAP header defining a trace level and trace stream ID.