

# Web Services Bootcamp

Bill Buchan - HADSL



- **Introduction**
- Web Services Overview
- Using Domino to provide Web Services
  - ▶ LotusScript in Domino v7 and v8
- Using Notes to consume Web Services
  - ▶ LotusScript in Notes 8
  - ▶ COM
  - ▶ Stubby in Notes 7
- Summary, Q+A



- I am Bill Buchan. I am:
  - ▶ A blogger - <http://www.billbuchan.com>
    - I was blogging when it was called 'Talking rubbish on the internet'.
    - Some things never change.
  - ▶ A principal of HADSL - <http://www.hadsl.com>
  - ▶ A Notes veteran of some 14 years
  - ▶ Old. Its my birthday today, so no loud noises...
  
- You are
  - ▶ Lotus Domino developers
  - ▶ Interested in starting or enhancing Web Services in your environment
  - ▶ Familiar with LotusScript, and perhaps Java



- This takes you from a low level in a subject to a good level of expertise
- Little existing knowledge is assumed
  - ▶ But we assume you are developers, and are familiar with LotusScript
- People have NOT paid to come to this session
  - ▶ But still, as a matter of courtesy, please switch off all mobile phones, etc
  - ▶ If your phone goes off, you will have to buy a drink for everyone in the room!
- Feel free to ask some questions as we go along
  - ▶ But I reserve the right to leave some answers till the end
- I will be doing live demonstrations
  - ▶ So feel free to laugh if they break!



- Introduction
- **Web Services Overview**
- Using Domino to provide Web Services
  - ▶ LotusScript in Domino v7 and v8
- Using Notes to consume Web Services
  - ▶ LotusScript in Notes 8
  - ▶ COM
  - ▶ Stubby in Notes 7
- Summary, Q+A



- Web services are:
  - ▶ A standard application to application protocol for exchanging structured data
    - Usually (but not always) using the http protocol
    - Usually (but not always) using XML
    - Usually we provide meta-information on what our service will do, using a language called WSDL (Web Service Description Language) – formatted in XML.
- Web services are:
  - Language independent
  - Platform independent
  - Data format representation independent
- But hopefully you all knew this already...



- The players:
  - ▶ The Web Service server
    - Can answer questions on what it can do
    - Can accept requests from Web Service clients
    - Can respond to requests from Web service clients
  - ▶ The Web Service client (or consumer)
    - Knows where the web service server is located
    - Knows the services that it requires
    - Knows how to prepare a Web Services Query
    - Knows how to interpret the returned data



- A typical web service session looks like:
  - ▶ Client to Web Service Server
    - “Tell me the answer to this question”
  - ▶ Web Service Server
    - “Here it is”
  
- Conclusion:
  - ▶ The clients drive the conversation
  - ▶ The Server cannot initiate a conversation with the client.



- The Web Services Protocol:
  - ▶ The client decides it needs to ask the server for some information
  - ▶ It constructs the relevant XML based web services query
  - ▶ It then connects to the Web Service Server
    - Pushing the web request up as an HTTP 'Post' request
    - It waits for a response
  - ▶ The Web service server then 'Posts' back an XML payload representing the answer
  - ▶ The client then unpacks and interprets the data.



- Not really.
  - ▶ Its a combination of web protocol and XML construction and unpacking
  - ▶ Most of the hard work is done for you by the various Web Service servers and Consumers
  - ▶ Its fairly easy to debug using the correct tools



- Fairly simple Domino Contact Database
  - ▶ A single 'Contact' form with some information
  - ▶ Not meant to be a 'real-world' example
  - ▶ Its here to demonstrate the techniques
  
- All example code is in the database
  - ▶ Simple to figure out and 'research'
  - ▶ Probably isnt best practice!



# Example Database: Contacts.nsf

LS2008 Web Services Bootcamp - Contacts\By Name - IBM Lotus Notes

File Edit View Create Actions Help

Workspace LS2008 Web Services Bootcam... X

Lotusphere 2008 LS2008 Web Services Bootcamp  
Bill Buchan  
hads!

Lotusphere 2008

New.. Goto.. Edit

Name	Contact
Barney Rubble	eMail: Barney.Rubble@bedrock.com Personal: 1 712 313 5555
Fred Flintstone	Mobile: 1 711 711 5555 eMail: fred@bedrock.com Personal: 1 721 211 5555

*LS2008 Web Services Bootcamp hosted on this computer in location: customers\LS2008\Contacts.nsf.  
You are logged in as: Bill S Buchan.*

Disconnected idm-demo1



## Contact: Barney Rubble

Contact:		Contact Details	
First Name:	『Barney』	Tel:	『1 712 313 5545』
Middle Initials:	『』	Mobile Tel:	『』
Last Name:	『Rubble』	eMail:	『Barney.Rubble@bedrock.com』
Full Name:	Barney Rubble	Home Tel:	『1 712 313 5555』
Notes:	『Small but perfectly formed』		



- I recommend SoapUI
  - ▶ Its free!
  - ▶ <http://www.soapui.org>
- It allows you to interrogate a web service
  - ▶ And build test case with real data



# SoapUI: Example Screenshot

The screenshot displays the SoapUI 1.7.1 application window. The main interface is divided into several sections:

- Projects Panel (Left):** Shows a tree view of projects. The 'BBPortal' project is expanded, showing a 'DominoSoapBinding' with several endpoints: GETAPPLICATIONTITLE, GETCATEGORY, GETUSERDETAILS, GETUSERS, GETUSERSBYNAME, GETVIEWS, and SEARCHFOR. A 'test' folder is also visible.
- Request Editor (Top Center):** Displays the SOAP request for the 'GETAPPLICATIONTITLE - Request 1' endpoint. The URL is 'http://bes1.jsector.net/servlet/Portal'. The XML content is:

```
<?xml version='1.0' encoding='UTF-8'>
<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/'>
  <soapenv:Header/>
  <soapenv:Body>
    <urn:GETAPPLICATIONTITLE soapenv:encoding='UTF-8'>
  </soapenv:Body>
</soapenv:Envelope>
```
- Response Editor (Right):** Displays the SOAP response for the same endpoint. The XML content is:

```
<?xml version='1.0' encoding='UTF-8'>
<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/'>
  <soapenv:Body>
    <ns1:GETAPPLICATIONTITLEResponse soapenv:encoding='UTF-8'>
      <GETAPPLICATIONTITLEReturn xsi:type='xsd:string'>
    </ns1:GETAPPLICATIONTITLEResponse>
  </soapenv:Body>
</soapenv:Envelope>
```
- Details Panel (Bottom):** Shows 'SOAP Request' and 'SOAP Response' tabs. Below these, it indicates 'Request Attachments (0)', 'HTTP Headers (0)', 'Response Attachments (0)', and 'HTTP Headers (6)'. The response time is '38043ms (579 bytes)'. A log window at the bottom shows system messages, including 'Missing folder [C:\Program Files\eviware\soapUI-1.7.1\bin\ext] for external libraries' and 'Loading workspace from [...]'. Log tabs for 'soapUI log', 'http log', and 'jetty log' are visible.

- Introduction
- Web Services Overview
- **Using Domino to provide Web Services**
  - ▶ LotusScript in Domino v7 and v8
- Using Notes to consume Web Services
  - ▶ LotusScript in Notes 8
  - ▶ COM
  - ▶ Stubby in Notes 7
- Summary, Q+A



- Three different methods:
  - ▶ Using LotusScript as a Web service
  - ▶ Coding a Java Servlet
  - ▶ Using an agent
- By far the simplest is to use a LotusScript Web Service
  - ▶ The majority of the work is done for you!
- Servlets should be avoided unless
  - ▶ You are running on Domino v6.5.x or older
  - ▶ You absolutely require persistence
- Agents are good for
  - ▶ Very low transaction services
  - ▶ Simple services
- We're only talking about using LotusScript



- Introduction
- Web Services Overview
- Using Domino to provide Web Services
  - ▶ **LotusScript in Domino v7 and v8**
- Using Notes to consume Web Services
  - ▶ LotusScript in Notes 8
  - ▶ COM
  - ▶ Stubby in Notes 7
- Summary, Q+A

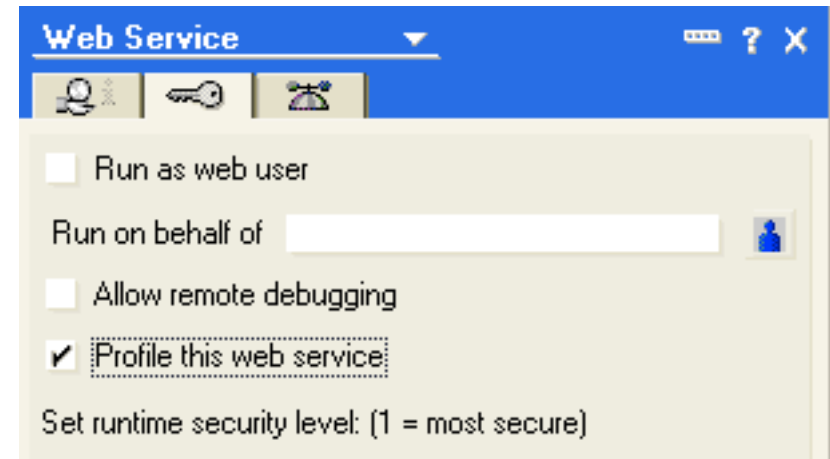
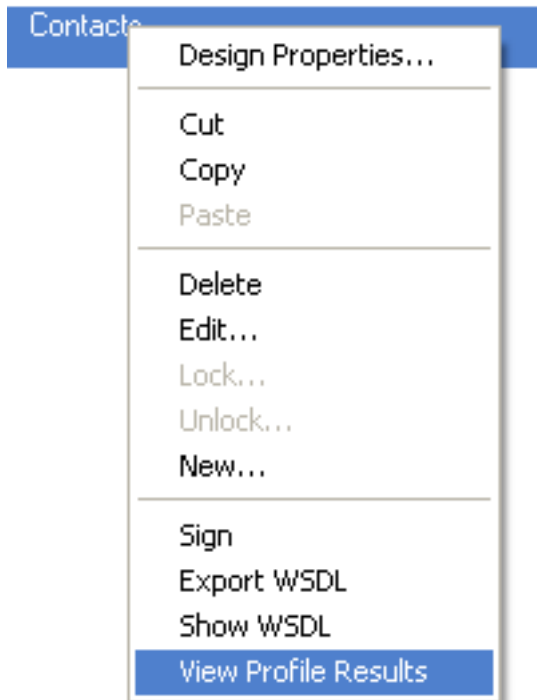


- Introduced in Lotus Domino 7
- Robust
- Code the web service using LotusScript
- Fairly high performance but
  - ▶ Remember: Lack of persistence between calls, so
    - Be aware of what your code is doing!
    - Use 'Agent Profiling' to see timings...



- LotusScript web services are very similar to LotusScript agents

- ▶ So you can profile them.
- ▶ Enable Profiling on the Web Services Properties tab



- ▶ To view the last run profile:
  - Right click on the Web Service
  - View Profile Results



- This shows how long the agent took to run, and how long the agent spent inside the Notes Object Interface
  - ▶ In our case, 761ms total, and 0ms within NOI
  - ▶ One call to CurrentDatabase, and one call to Title.

## Contacts Profile

15/12/2007 23:42:43 GMT

Elapsed time: 761 msec

Methods profiled: 2

Total measured time: 0 msec

Class	Method	Operation	Calls	Time
Database	Title	Get	1	0
Session	CurrentDatabase	Get	1	0

- ▶ Obviously, this wasn't a very interesting web service call...



- Spend time thinking about how your consumer will use this web service.
  - ▶ More time thinking, less time coding
  - ▶ You do NOT want to change this web service interface once its published
    - Instead - roll out a new web service
  - ▶ Think through client use cases
  - ▶ Remember - you can use LotusScript to perform complex business rules, and provide a summary of data to your consumer
    - Don't assume the consumer can perform a lot of data manipulation
      - BlackBerry smartphones / Windows Mobile ?
- Best to put complex code in a script library, and just surface it via a web service
  - It means you can put a normal agent 'test harness' around it and debug it



- In our case we wish to:
  - ▶ Be able to list all users
    - By Name
    - By Department
  - ▶ Be able to get details on a particular user
  - ▶ Be able to search for users
- Do we
  - ▶ Pass all fields back as web service items?
  - ▶ Pass back an array of fields?
- In this case, we pass back Web Service Items
  - ▶ Assume data mapping knowledge and responsibility.



- We design our web service interface within a 'Class' in LotusScript.
  - ▶ At this point, you probably wished that you studied the Object Orientated LotusScript sessions I used to give...
  - ▶ Good example presentations on my personal blog presentations page
    - <http://www.billbuchan.com/web.nsf/plinks/BBUN6MQECQ.htm>
- Our web service will expose all 'public' methods and properties
  - ▶ We do not have to put all the code in one 'class'
  - ▶ Beware of extending existing classes - you might expose more than you want!
    - Beware over-exposure...



- LotusScript does NOT allow functions to return Arrays.
  - ▶ This is a language limitation, not a Web Service Limitation
  - ▶ It can return simple types
  - ▶ A variant (but since web services don't support `em, we cant use `em)
  - ▶ Instances of other classes...
- We need to use arrays to return results
  - ▶ For instance, we need to return an array of Zero or more entries to list our users.
  - ▶ How do we do that?
- We can define a class and return an instance of that class...



- We can define a class called 'ReturnArray' which has a single array as a public member.
  - ▶ The constructor initialises the array so that it has one blank entry.
- We can then populate this array with one or more entries.
- LotusScript Web services only
- supports single dimension arrays.
- Nd7 supports arrays with one element or more
- nd8 supports 'empty' arrays.

```
Class returnArray  
  
    Public S() As String  
  
    Sub new  
  
        Redim S(0)  
  
        S(0) = ""  
  
    End Sub  
  
End Class
```



- We shall return our Person contact record using a class.
  - ▶ We DON'T have a constructor
  - ▶ All elements are Public - so it'll be exposed to the Web Service as data.
  - ▶ Our property names will be used by the web service as field names
    - Since LotusScript is case insensitive, the web service field names will be UPPERCASE. Which looks funky in other platforms.

## Class Person

```
Public FirstName As String  
Public MiddleInitials As String  
Public LastName As String  
Public FullName As String
```

```
Public Telephone As String  
Public Cellphone As String  
Public eMail As String  
Public HomePhone As String  
Public Department as String
```

```
Public Notes As String
```

```
End Class
```



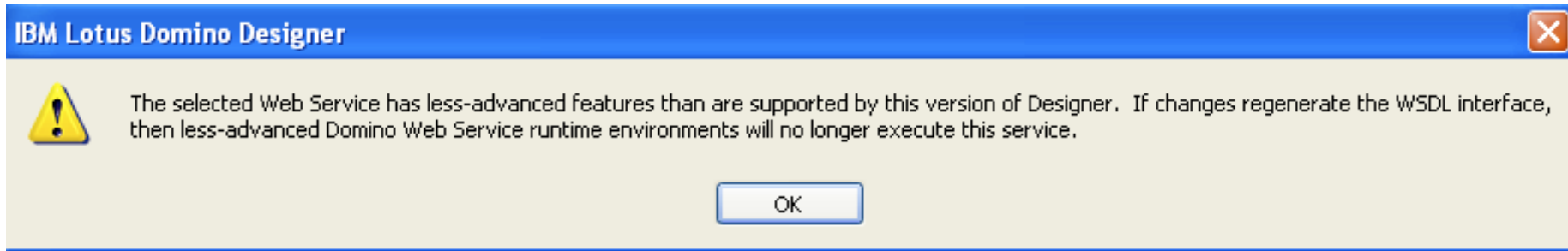
- Finally! A Demo!



- We need to add:
  - ▶ Logging. We need to know when it works, and importantly, when it does not
  - ▶ Error Handling. We need to pass errors back to the consumer
  - ▶ Security. We might want to harden this service somewhat
- We committed the following sins:
  - ▶ We put all the code in the web service itself, making it hard to test. Best to consider embedding most of the code in script libraries
  - ▶ We've tightly bound the data structures in our data to our web service
    - If we add more fields to our contact record - we shall have to change the web service. And therefore, all our clients will have to be checked and/or recompiled.
    - If the data set is likely to change, construct a more flexible data-mapping schema



- Web services built or compiled in Notes 8 Designer WONT run on ND7 anymore!



So be careful about what web services you host on which servers, and which version of designer you use.

- ▶ Just bear this version difference in the same manner as all other version differences.



- In ND7, the class used for the web service has to reside in the web service design element. In ND8, this can be in a script library.
  - ▶ This means that for complex services, you can place all the business code in a script library, allowing simple construction of a test harness.
- ND8 now supports code in Java Libraries
- ND8 supports more error handling SOAP elements
- ND8 supports 'empty' arrays - nd7 does not



- Demo: Lets go test this service with SoapUI



- WSDL - Web Services Description Language
  - ▶ Its an XML document
  - ▶ You can query a web service for its WSDL,
  - ▶ You build a web service consumer from the WSDL
- Lotus Domino LotusScript Web Services WSDL
  - ▶ Is automatically generated for you
  - ▶ You can query a LotusScript Web Service by passing a URL:
    - `http://<host>/<directory>/<database>/<web service>?WSDL`
    - It will then return an XML document showing the WSDL



- The first section outlines the data objects we shall pass from this service
  - ▶ You can see our returnArray & Person objects
- Note that our variables default to Uppercase - this is a side-effect of LotusScript being a non-case sensitive language

```
<wsdl:definitions targetNamespace="urn:DefaultNamespace">
- <wsdl:types>
- <schema targetNamespace="urn:DefaultNamespace">
  <element name="DEPTNAME" type="xsd:string"/>
- <complexType name="RETURNARRAY">
  - <sequence>
    <element maxOccurs="unbounded" minOccurs="0" name="S" type="xsd:string"/>
  </sequence>
</complexType>
<element name="LISTUSERSBYDEPARTMENTReturn" type="impl:RETURNARRAY"/>
<element name="LISTUSERSReturn" type="impl:RETURNARRAY"/>
<element name="CONTACTNAME" type="xsd:string"/>
- <complexType name="PERSON">
  - <sequence>
    <element name="FIRSTNAME" type="xsd:string"/>
    <element name="MIDDLEINITIALS" type="xsd:string"/>
    <element name="LASTNAME" type="xsd:string"/>
    <element name="FULLNAME" type="xsd:string"/>
    <element name="TELEPHONE" type="xsd:string"/>
    <element name="CELLPHONE" type="xsd:string"/>
    <element name="EMAIL" type="xsd:string"/>
    <element name="HOMEPHONE" type="xsd:string"/>
    <element name="DEPARTMENT" type="xsd:string"/>
    <element name="NOTES" type="xsd:string"/>
    <element name="ERRORSTRING" type="xsd:string"/>
  </sequence>
</complexType>
<element name="GETCONTACTDETAILSReturn" type="impl:PERSON"/>
</schema>
</wsdl:types>
```

- The second section of our WSDL deals with the Web Services Messages that will be sent to the Web service, and the types that get returned.

```
--  
<wsdl:message name="LISTUSERSBYDEPARTMENTRequest">  
  <wsdl:part element="intf:DEPTNAME" name="DEPTNAME"/>  
</wsdl:message>  
<wsdl:message name="GETCONTACTDETAILSResponse">  
  <wsdl:part element="intf:GETCONTACTDETAILSReturn"  
    name="GETCONTACTDETAILSReturn"/>  
</wsdl:message>  
<wsdl:message name="LISTUSERSResponse">  
  <wsdl:part element="intf:LISTUSERSReturn" name="LISTUSERSReturn"/>  
</wsdl:message>  
<wsdl:message name="LISTUSERSBYDEPARTMENTResponse">  
  <wsdl:part element="intf:LISTUSERSBYDEPARTMENTReturn"  
    name="LISTUSERSBYDEPARTMENTReturn"/>  
</wsdl:message>  
<wsdl:message name="LISTUSERSRequest"/>  
<wsdl:message name="GETCONTACTDETAILSRequest">  
  <wsdl:part element="intf:CONTACTNAME" name="CONTACTNAME"/>  
</wsdl:message>
```



- The port section associates input messages - function calls - with their return types.

```
<wsdl:portType name="Contacts">
- <wsdl:operation name="LISTUSERSBYDEPARTMENT">
  <wsdl:input message="intf:LISTUSERSBYDEPARTMENTRequest"
  name="LISTUSERSBYDEPARTMENTRequest"/>
  <wsdl:output message="intf:LISTUSERSBYDEPARTMENTResponse"
  name="LISTUSERSBYDEPARTMENTResponse"/>
</wsdl:operation>
- <wsdl:operation name="LISTUSERS">
  <wsdl:input message="intf:LISTUSERSRequest" name="LISTUSERSRequest"/>
  <wsdl:output message="intf:LISTUSERSResponse" name="LISTUSERSResponse"/>
</wsdl:operation>
- <wsdl:operation name="GETCONTACTDETAILS">
  <wsdl:input message="intf:GETCONTACTDETAILSRequest"
  name="GETCONTACTDETAILSRequest"/>
  <wsdl:output message="intf:GETCONTACTDETAILSResponse"
  name="GETCONTACTDETAILSResponse"/>
</wsdl:operation>
</wsdl:portType>
```



- The binding section defines all input and output operations

```
<wsdl:binding name="DominoSoapBinding" type="intf:Contacts">
  <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  - <wsdl:operation name="LISTUSERSBYDEPARTMENT">
    <wsdlsoap:operation soapAction="LISTUSERSBYDEPARTMENT"/>
    - <wsdl:input name="LISTUSERSBYDEPARTMENTRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    - <wsdl:output name="LISTUSERSBYDEPARTMENTResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  - <wsdl:operation name="LISTUSERS">
    <wsdlsoap:operation soapAction="LISTUSERS"/>
    - <wsdl:input name="LISTUSERSRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    - <wsdl:output name="LISTUSERSResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  - <wsdl:operation name="GETCONTACTDETAILS">
    <wsdlsoap:operation soapAction="GETCONTACTDETAILS"/>
    - <wsdl:input name="GETCONTACTDETAILSRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    - <wsdl:output name="GETCONTACTDETAILSResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```



## ■ The Service section

- ▶ Connects the service to the binding
- ▶ Gives a fully formed URL for the service
- ▶ Note that it uses the current servers' name in the URL
  - This may be an issue if you're masking the URL

```
<wsdl:service name="ContactsService">  
- <wsdl:port binding="intf:DominoSoapBinding" name="Domino">  
  <wsdlsoap:address  
    location="http://fdm-demo1:80/customers/ls2008/contacts.nsf/Contacts?OpenWebService"/>  
  </wsdl:port>  
</wsdl:service>
```



- Introduction
- Web Services Overview
- Using Domino to provide Web Services
  - ▶ LotusScript in Domino v7 and v8
- **Using Notes to consume Web Services**
  - ▶ LotusScript in Notes 8
  - ▶ COM
  - ▶ Stubby in Notes 7
- Summary, Q+A



- Why consume web services?
  - ▶ To gain access to information in other systems
  - ▶ To prevent the synchronisation and replication of external data sources to Notes
  - ▶ Real time, up to date data lookup
- Where?
  - ▶ We could use a scheduled Java agent in Domino to interrogate external services
    - No UI means you have to monitor and check
  - ▶ We could use client-driven code on the client workstation
    - Be aware of network issues between the clients and the remote service
      - Large corporate intranets are rarely stable and predictable
      - Predict & accommodate remote service outage
    - Any issues are immediately visible to the users
      - Empowering or Confusing?



- Security
  - ▶ Remote web services may require
    - Username/password login
    - Encrypted username/password pairs in data
    - SSL.
  
- Things change
  - ▶ You may have to change your web service consumer to accommodate this
  - ▶ The remote web service may move
  - ▶ When designing, don't hard-code...



- In the following sections,
  - ▶ We shall use the web services we constructed earlier as a target
  - ▶ We wont use any security features for simplicity
  - ▶ We're coding to demonstrate technique, not best practices
    - So by all means reuse this code, but make it production-strength
      - Add Error handling
      - Remove hard-coding
      - Add Logging



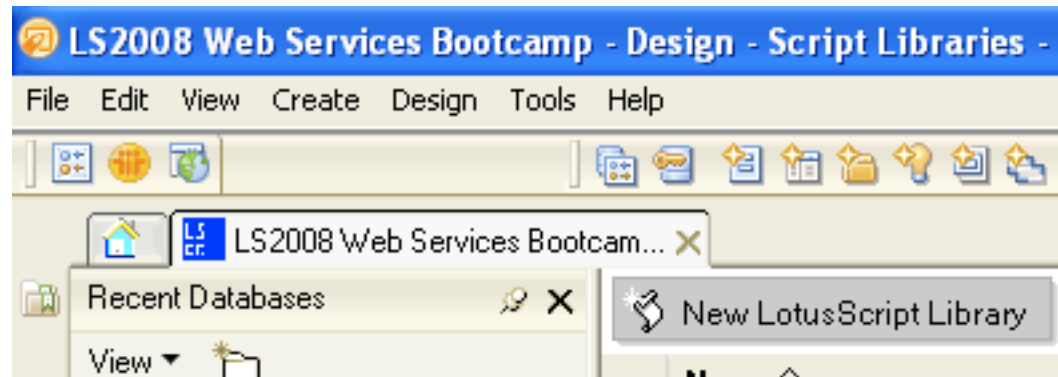
- Introduction
- Web Services Overview
- Using Domino to provide Web Services
  - ▶ LotusScript in Domino v7 and v8
- Using Notes to consume Web Services
  - ▶ **LotusScript in Notes 8**
  - ▶ COM
  - ▶ Stubby in Notes 7
- Summary, Q+A



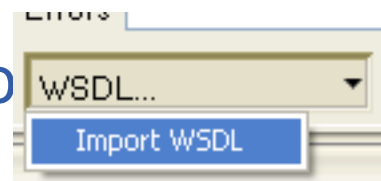
- Summary:
  - ▶ Notes 8
  - ▶ Its very simple
  - ▶ It works in LotusScript and in Java
  - ▶ It does a lot of work for you
  
- How do I construct a Web Service Consumer in LotusScript?
  - ▶ Create a new, empty script library
  - ▶ Click on the Import WSDL button
  - ▶ It creates a Class definition
  - ▶ You can then “use” the script library and class.



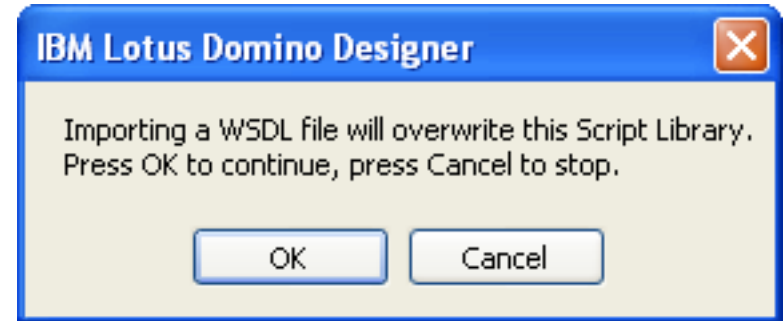
- Create a new LotusScript Script Library:



- Click on the WSDL button



- It needs a WSDL file. What's that?
  - ▶ You can open the WSDL definition for your web service in your browser
  - ▶ Use View Source, and save that as a WSDL file.
  - ▶ Select it...
- Designer will then construct helper classes and a main class which we can then call
- That's it!



Sub Initialize

Dim C As New Contacts

Dim V As Variant  
Set V = C.listUsers()

Forall thisUser In V.S  
    Print "Found User: " + thisUser  
End Forall

End Sub



- Lets go build our LotusScript consumer LIVE...



- The entire LotusScript library is taken up with the computed Web service definition. Best not to make changes.
  - ▶ Next time its refreshed, you will lose those changes!
- If you change the web service, you must remember to update the Web Services consumer.
  - ▶ It doesn't take long - so don't forget it.
  - ▶ And when you change the web service consumer script library, you must recompile all the LotusScript that relies on it.
    - 'Tools, Recompile LotusScript' is your friend



- It requires the Notes 8 client to run on:
  - ▶ We as Application Developers deploy solutions on users' target notes clients
  - ▶ Notes 8 may not yet be adopted in your environment
  - ▶ This may help drive the upgrade process
  
- What if you need Web Service consumption now, and are not yet on Notes 8?
  - ▶ The following sections will answer your questions...



- Introduction
- Web Services Overview
- Using Domino to provide Web Services
  - ▶ LotusScript in Domino v7 and v8
- Using Notes to consume Web Services
  - ▶ LotusScript in Notes 8
  - ▶ **COM**
  - ▶ Stubby in Notes 7
- Summary, Q+A



- There are applications which enable you to perform web service consumer work directly from LotusScript
  - ▶ We shall focus on the MS Soap Toolkit
  - ▶ Its not the only one - many others exist
- Very Quick and Very Dirty
  - ▶ Its platform specific
  - ▶ You need to install a DLL on each workstation
- When would you use this?
  - ▶ Building test harnesses
  - ▶ Perhaps on low user-count web service applications



- So what does the code look like?

```
Dim Client As Variant  
Set Client = CreateObject("MSSOAP.SoapClient")
```

```
'Initialize connection to the Web Service  
Call Client.mssoapinit ( url )
```

```
Dim result As String
```

```
result = Client.getApplicationName()
```

```
Msgbox "Application Name is: " + result
```



- The MS Soap web service consumer:
  - ▶ Is Platform specific - windows only
  - ▶ Requires installation
  - ▶ Is no longer supported
  - ▶ Doesn't handle complex objects well
    - Arrays and Class Objects are returned as COM objects
    - Very opaque - difficult to work out how they work
  - ▶ Doesn't sound very useful
    - But it can help quickly build test harnesses
- A more useful COM based SOAP consumer:
  - ▶ Microsoft XML HTTP object...



- Its far more low level than MS SOAP
  - ▶ You have to construct the HTTP header, SOAP payload
  - ▶ You get back raw text, which you can pass to an XML parser
- Its bundled by default with various Microsoft Packages
  - ▶ You might find that its already deployed in your environment
  - ▶ Various versions exist



- Introduction
- Web Services Overview
- Using Domino to provide Web Services
  - ▶ LotusScript in Domino v7 and v8
  - ▶ Java using a servlet in Domino 5 upwards
  - ▶ Using an agent
  - ▶ REST
- Using Notes to consume Web Services
  - ▶ LotusScript in Notes 8
  - ▶ COM
  - ▶ **Stubby in Notes 7**
- Summary, Q+A



- It:
  - ▶ Is a project on <http://OpenNtf.org>
  - ▶ Generates the relevant java stub files to allow you to consume java web services
  - ▶ runs on the notes client
  - ▶ Only runs on Notes 7
    - So it's a very tactical solution...
- Use the helper form to create the necessary stub files:



- Enter the URL for the web service and click 'Generate Stub Files'

## Step 1

Please enter the location of the WSDL file that we will use to generate the stub files. This can be either a URL (like <http://localhost/wstest.nsf/MyWebService?WSDL>) or a local file reference (like `c:\temp\wstest.wsdl`). Please note that web service URLs that require authentication will not work properly – you'll need to copy those WSDL files to your local machine and access them from the hard drive.

**WSDL File:**

## Step 2

Click the button to generate the Axis "stub" files based on the WSDL file above (note that this document must be in Edit mode to see the button):

Check this box to delete the temporary files after stub files have been generated

## Step 3

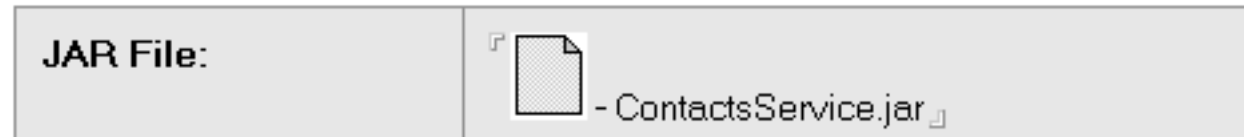
If there were errors, you will see them here.

**Stub File Errors:**

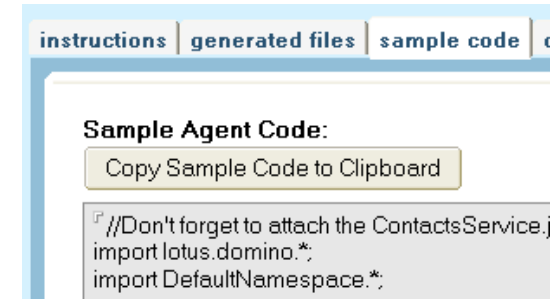


- Save the JAR file to your local hard drive

These files were generated when the "Generate Stub Files" button was clicked



- Copy the same code to the clipboard



- Go to your database, create a new java agent
  - ▶ Edit Project, and attach the JAR file
  - ▶ Paste the sample code into the Java agent
  - ▶ Extend the class to use the web service methods



- So far this has created a platform independent Java agent.
  - ▶ But we're interested in LotusScript!
- We have two easy ways of interfacing Java and LotusScript agents
  - ▶ LS2J
  - ▶ Notes agents can call Notes agents...



- We can write a LotusScript agent
  - ▶ Write a document with some parameter information
  - ▶ Call a Java agent, and pass the NoteID of the document
  - ▶ In the Java agent
    - Read the parameters
    - Write the results
    - Save the Document
  - ▶ Back in the LotusScript agent, re-open the same NoteID
    - Read the results



- Demo: Lets call a stubby web service from LotusScript.



- Introduction
- Web Services Overview
- Using Domino to provide Web Services
  - ▶ LotusScript in Domino v7 and v8
  - ▶ Java using a servlet in Domino 5 upwards
  - ▶ Using an agent
  - ▶ REST
- Using Notes to consume Web Services
  - ▶ LotusScript in Notes 8
  - ▶ COM
  - ▶ Stubby in Notes 7
- **Summary, Q+A**



- Web Services
  - ▶ Are pretty straightforward
  - ▶ An important part of your armory
  - ▶ Help break down barriers between Domino and other systems
  - ▶ Help prevent data duplication in your environment
  
- By now, you know enough to make a real difference



- Please fill out your evaluations:
  - ▶ They are taken extremely seriously, and form the basis for next years speaker list..
- Thank you for your time today
  
- Bill Buchan
  - ▶ <http://www.billbuchan.com>
  - ▶ <http://www.hadsl.com>

