V22.0480-004
Web Services Architecture and Programming

Lecture 14
WSDL
Announcements

- Lab 4 available from the web site
  - due back November 12th, (Wednesday)
Web services allow programmatic interaction …

… by relying upon standard protocols for

- Discovering services with desired characteristics
  - UDDI (next lecture)

- Determining the operations offered by the service
  - WSDL (this lecture)

- Invoking the desired operation
  - SOAP (previous lecture)
Web Services Description Language

WSDL describes four aspects of the service

• Interface information
  – What are its publicly available functions?

• Data type information
  – What information is required/produced by message requests and responses?

• Binding information
  – Which transport protocols can be used?

• Address information
  – Where is the service located?

• A contract between service provider and consumer
  – Platform and language independent (unlike XML-RPC, .NET Remoting)
  – Tools automate process of locating, invoking web service functionality
WSDL Specification

Two key concepts

- **Services** are a collections of network endpoints, or **ports**
- Separation between the **abstract** definition of an endpoint, and its **concrete** network deployment

Realized in terms of

- **Messages**: Description of data being exchanged
- **Port types**: Collections of operations
- **Binding**: Concrete protocol and data format specifications for a particular port type
- **Port**: Binding + network address
- **Service**: Collection of ports
WSDL Elements

- `<definitions> ... </definitions>`
  - Root element of WSDL specification
  - Defines name of the web service, multiple namespaces

- `<types>...</types>`
  - Describes types used between the client and server
  - Default: XML Schema specification

- `<message>...</message>`
  - A one-way message, containing zero or more `<part>...</part>` elements

- `<portType>...</portType>`
  - Combines multiple messages to form operation(s)

- `<binding>...</binding>`
  - Specifics of how service will be implemented on wire, SOAP specifics, ...

- `<service>...</service>`
  - One or more ports, each with its own network address
WSDL Document – Example

- Example from netserver1.pdsg.cs.nyu.edu: SumAndDifference
  
  [ http://netserver1.pdsg.cs.nyu.edu/VSDev/vijayk/WebServices/
    SumAndDifference/SumAndDifference.asmx?WSDL ]

- Available at

  http://netserver1.pdsg.cs.nyu.edu/VSDev/Public/vijayk/
  Lect14-Files/SumAndDifference.wsdl
WSDL – <definitions> … </definitions>

```xml
<definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:s0="http://netserver1.pdsg.cs.nyu.edu/vijayk/webservices/
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/
    targetNamespace="http://netserver1.pdsg.cs.nyu.edu/vijayk/webservices/
    xmlns="http://schemas.xmlsoap.org/wsdl/">
    
</definitions>
```

- Namespace for this document
- Default namespace (for elements without a namespace prefix)

- Can specify a name for the service
  - One of the [WebService(…)] options in Visual Studio.NET
- Also defines additional namespaces used in the rest of the document
WSDL – <types> … </types>

• Defines the types used in messages
• WSDL itself is not tied to any specific typing system, but it uses the XML Schema specification as its default choice
  – Built-in types, simple and complex types
WSDL – `<message> … </message>`

```xml
<message name="ComputeSoapIn">
    <part name="parameters" element="s0:Compute" />
</message>
<message name="ComputeSoapOut">
    <part name="parameters" element="s0:ComputeResponse" />
</message>
```

- Message attribute specifies name of message
- `<part> … </part>` element specifies parameters
  - By convention, SOAP RPC messages have only one `<part> … </part>` element, a structure containing the real parameters
WSDL – `<portType> ... </portType>`

```xml
<portType name="MainClassSoap">
    <operation name="Compute">
        <input message="s0:ComputeSoapIn" />
        <output message="s0:ComputeSoapOut" />
    </operation>
</portType>
```

- An abstract collection of **operation(s)**
- Each operation consists of a **pattern** of messages
  - Message names must be namespace-qualified

- Four basic patterns
  - One-way
  - Request-response
  - Solicit-response
  - Notification
### WSDL – portType `<operation> ... </operation>`

<table>
<thead>
<tr>
<th>Type</th>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-way</td>
<td><img src="image" alt="One-way Diagram" /></td>
</tr>
<tr>
<td>Request-response</td>
<td><img src="image" alt="Request-response Diagram" /></td>
</tr>
<tr>
<td>Solicit-response</td>
<td><img src="image" alt="Solicit-response Diagram" /></td>
</tr>
<tr>
<td>Notification</td>
<td><img src="image" alt="Notification Diagram" /></td>
</tr>
</tbody>
</table>
WSDL – <binding> … </binding>

```xml
<binding name="MainClassSoap" type="s0:MainClassSoap">  
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"  
  style="document" /> 
  <operation name="Compute">  
    <soap:operation  
    soapAction="http://netserver1.pdsg.cs.nyu.edu/vijayk/webservices/Co  
    mpute" style="document" />  
    <input>  
      <soap:body use="literal" />  
    </input>  
    <output>  
      <soap:body use="literal" />  
    </output>  
  </operation> 
</binding>
```

- Type attribute references a portType defined earlier
- Provides information about how Compute operation messages are transported over the Internet
WSDL – SOAP Binding

• Built-in extensions to allow expression of SOAP-specific details
  – SOAP headers, encoding styles
  – SOAPAction HTTP header (identifies the service)

• `<soap:binding>`
  – Indicates binding will be made available via SOAP
  – `style` attribute indicates message format
    • `document`: simple XML documents (.NET preference)
    • `rpc`: additional wrapper element indicating the function name

• `<soap:operation>`
  – Indicates binding of a specific operation to a specific SOAP implementation (SOAPAction header)

• `<soap:body>`
  – For each operation, specifies details of the input/output messages
    • Encoding, header blocks, headerfault, fault, …
WSDL – `<service>` … `</service>`

```xml
<service name="MainClass">
    <port name="MainClassSoap" binding="s0:MainClassSoap">
        <soap:address
            location="http://216.165.111.6/VSDev/vijayk/WebServices/SumAndDifference/SumAndDifference.asmx" />
    </port>
</service>
```

- Service element specifies location of one or more ports
  - `binding` denotes portType (hence, operations, messages, types)
  - `location` provides info about where service is accessible
    - `soap:address` value goes into HTTP POST header
    - Used by IIS to route SOAP requests

- Can optionally contain human-readable documentation describing the service
  - An additional option to the `[WebService(…)]` decoration
SOAP Encoding of the Compute Operation (Input)

POST /VSDev/vijayk/WebServices/SumAndDifference/SumAndDifference.asmx
HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol 1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://netserver1.pdsg.cs.nyu.edu/vijayk/webservices/Compute"
Authorization: Negotiate
Content-Length: 343
Expect: 100-continue
Host: localhost:9000

<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><Compute
xmlns="http://netserver1.pdsg.cs.nyu.edu/vijayk/webservices/"><a>8</a><b>5</b></Compute></soap:Body></soap:Envelope>
Production and Consumption of WSDL in VS.NET

Service side
• WSDL description can be retrieved by navigating to the service URL appended with “?WSDL”
  – Functionality built-in when assembly is produced

Client side
• Generate proxy from WSDL description
  – Implicitly by adding a “Web Reference”
  – Explicitly by invoking the command-line tool “wsdl”

```cmd
wsdl /n:Client.SumAndDifference /o:proxy.cs SumAndDifference.wSDL
```

http://netserver1.pdsg.cs.nyu.edu/VSDev/Public/vijayk/Lect14-Files/{proxy.cs.txt, SumAndDifference.wsd1}