

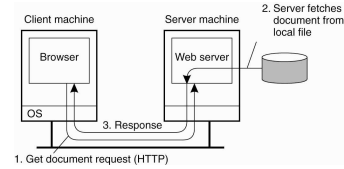
Web Services

12 January 2010
Lecture 13

Slide Credits: Maarten van Steen, Zack Ives

Distributed Web-Based Systems

Essence: The WWW is a huge client-server system with millions of servers; each server hosting thousands of **hyperlinked** documents:



- Documents are generally represented in text (plain text, HTML, XML)
- Alternative types: images, audio, video, but also applications (PDF, PS)
- Documents may contain scripts that are executed by the client-side software

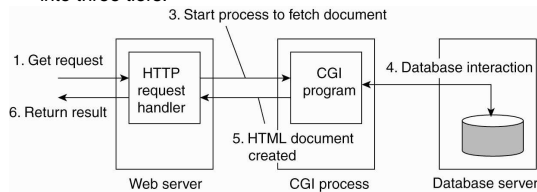
January 12, 2010

ISE 431: Distributed Information Systems

2

Multi-tiered Architectures

Observation: Already very soon, Web sites were organized into three tiers:



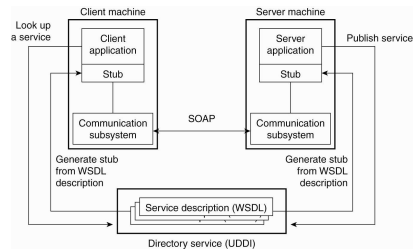
January 12, 2010

ISE 431: Distributed Information Systems

3

Web Services

Observation: At a certain point, people started recognizing that it was more than just user ↔ site interaction: sites could offer **services** to other sites → **standardization** is then badly needed.



January 12, 2010

ISE 431: Distributed Information Systems

4

Web Services

- Goal: provide an infrastructure for connecting components, building applications in a way similar to hyperlinks between data
- It's another distributed computing platform for the Web
 - Goal: Internet-scale, language-independent, upwards-compatible where possible
- This one is based on many familiar concepts
 - Standard protocols: HTTP
 - Standard marshalling formats: XML-based, XML Schemas
 - All new data formats are XML-based

January 12, 2010

ISE 431: Distributed Information Systems

5

Three Parts to Web Services

1. "Wire" / messaging protocols
 - Data encodings, RPC calls or document passing, etc.
2. Describing what goes on the wire
 - Schemas for the data
3. "Service discovery"
 - Means of finding web services

January 12, 2010

ISE 431: Distributed Information Systems

6

A Note on XML

Observation: XML has the advantage of allowing **selfdescribing documents**. **Full stop** (i.e., it introduces performance problems and is **not** meant to be read by human beings)

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
      <n:priority>k</n:priority>
      <n:expires>2001-06-22T14:00:00-05:00</n:expires>
    </n:alertcontrol>
  </env:Header>
  <env:Body>
    <m:alert xmlns:m="http://example.org/alert">
      <m:msg>Pick up Mary at school at 2pm</m:msg>
    </m:alert>
  </env:Body>
</env:Envelope>
```

January 12, 2010

ISE 431: Distributed Information Systems

7

Naming: URL

URL: Uniform Resource Locator tells how and where to access a resource.

Scheme	Host name	Port	Pathname
http	www.cs.vu.nl		/home/steen/box
(A)			
http	www.cs.vu.nl	80	/home/steen/box
(B)			
http	130.37.24.11	80	/home/steen/box
(C)			

Examples:

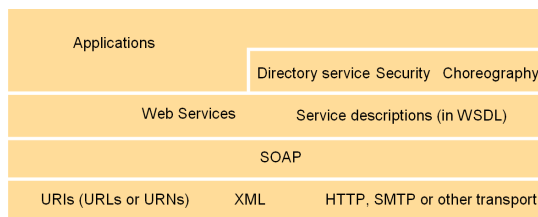
Name	Used for	Example
http	HTTP	http://www.cs.vu.nl/80/globe
mailto	E-mail	mailto:steen@cs.vu.nl
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	file://edu/book/work/chp/11/11
data	Inline data	data:text/plain;charset=iso-8859-7,%e1%e2%e3
telnet	Remote login	telnet://its.cs.vu.nl
tel	Telephone	tel:+31201234567
modem	Modem	modem:+31201234567:type=v32

January 12, 2010

ISE 431: Distributed Information Systems

8

Protocol Stack

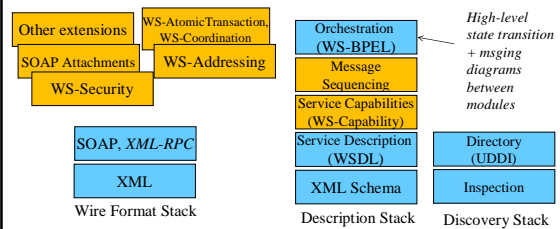


January 12, 2010

ISE 431: Distributed Information Systems

9

The Protocol Stacks of Web Services



Enhanced + expanded from a figure from IBM's "Web Services Insider", <http://www-106.ibm.com/developerworks/webservices/library/ws-ref2/>

January 12, 2010

ISE 431: Distributed Information Systems

10

SOAP

Simple Object Access Protocol: Based on XML, this is the standard protocol for communication between Web services.

- SOAP is **bound** to an underlying protocol (i.e., it is **not** independent from its **carrier**)
- **Conversational exchange style:** Send a **document** one way, get a filled-in response back.
- **RPC-style exchange:** Used to invoke a **Web service**.

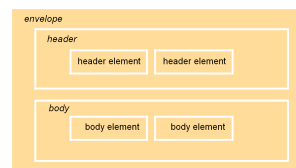
January 12, 2010

ISE 431: Distributed Information Systems

11

Messaging Protocol: SOAP

- Simple Object Access Protocol: XML-based format for passing parameters
 - Has a SOAP header and body inside an *envelope*
 - As a defined HTTP binding (POST with *content-type* of *application/soap+xml*)
 - A companion SOAP Attachments encapsulates other (MIME) data
- The header defines information about processing: encoding, signatures, etc.
 - It's extensible, and there's a special attribute called *mustUnderstand* that is attached to elements that *must* be supported by the callee
- The body defines the actual application-defined data



January 12, 2010

ISE 431: Distributed Information Systems

12

A SOAP Envelope

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Header>
    <t:Transaction xmlns:t="www.mytrans.com" SOAP-
      ENV:mustUnderstand="1" />
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <m:PlaceOrder xmlns:m="www.somewhere/there">
      <orderno xsi:type="xsd:string">12</orderno>
    </m:PlaceOrder>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

January 12, 2010

ISE 431: Distributed Information Systems

13

Making a SOAP Call (1/2)

- To execute a call to service PlaceOrder:

```
POST /PlaceOrder HTTP/1.1
Host: my.server.com
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnn
```

```
<SOAP-ENV:Envelope>
...
</SOAP-ENV:Envelope>
```

January 12, 2010

ISE 431: Distributed Information Systems

14

Making a SOAP Call (2/2)

```
POST /examples/stringer
Host: www.cdk4.net
Content-Type: application/soap+xml/
Action: http://www.cdk4.net/examples/stringer#exchange
```

Endpoint

Action

```
<env:envelope xmlns:env=namespace URI for SOAP
  envelope>
<env:header> </env:header>
<env:body> </env:body>
</env:envelope>
```

January 12, 2010

ISE 431: Distributed Information Systems

15

SOAP Return Values

- If successful, the SOAP response will generally be another SOAP message with the return data values, much like the request
- If failure, the contents of the SOAP envelop will generally be a Fault message, along the lines of:

```
<SOAP-ENV:Body>
  <SOAP-ENV:Fault xmlns="mynamespace">
    <faultcode>SOAP-ENV:Client</faultcode>
    <faultstring>Could not parse message</faultstring>
  ...
```

January 12, 2010

ISE 431: Distributed Information Systems

16

How Do We Declare Functions?

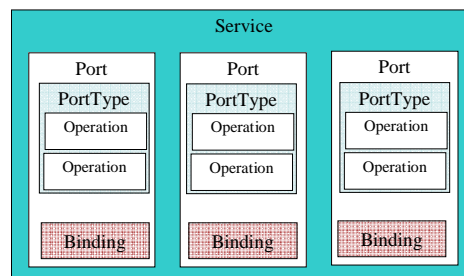
- WSDL is the interface definition language for web services
 - Defines notions of protocol bindings, ports, and services
 - Generally describes data types using XML Schema
- In CORBA, this is called an IDL (we didn't talk about this)
- In Java, the interface uses the same language as the Java code

January 12, 2010

ISE 431: Distributed Information Systems

17

A WSDL Service



January 12, 2010

ISE 431: Distributed Information Systems

18

Web Service Terminology

- Service: the entire Web Service
- Port: maps a set of port types to a transport binding (a protocol, frequently SOAP, COM, CORBA, ...)
- Port Type: abstract grouping of operations, i.e. a class
- Operation: the type of operation – request/response, one-way
 - Input message and output message; maybe also fault message
- Types: the XML Schema type definitions

January 12, 2010

ISE 431: Distributed Information Systems

:19

Example WSDL

```
<service name="POService">
  <port binding="my:POBinding">
    <soap:address location="http://yyy:9000/POSvc"/>
  </port>
</service>
<binding xmlns:my="..." name="POBinding">
  <soap:binding style="rpc" transport="http://www.w3.org/2001/..." />
  <operation name="POOrder">
    <soap:operation soapAction="POService/POBinding" style="rpc" />
    <input name="POOrder">
      <soap:body use="literal" ... namespace="POService" .../>
    </input>
    <output name="POOrderResult">
      <soap:body use="literal" ... namespace="POService" .../>
    </output>
  </operation>
</binding>
```

January 12, 2010

ISE 431: Distributed Information Systems

:20

JAX-RPC: Java and Web Services

- To write JAX-RPC web service “endpoint”, you need two parts:
 - An endpoint interface – this is basically like the IDL statement
 - An implementation class – your actual code

```
public interface BookQuote extends java.rmi.Remote {
  public float getBookPrice(String isbn) throws java.rmi.RemoteException;
}
public class BookQuote_Impl_1 implements BookQuote {
  public float getBookPrice(String isbn) { return 3.22; }
}
```

January 12, 2010

ISE 431: Distributed Information Systems

:21

Different Options for Calling

- The conventional approach is to generate a stub, as in the RPC model described earlier
- You can also *dynamically* generate the call to the remote interface, e.g., by looking up an interesting function to call
- Finally, the “DII” (Dynamic Instance Invocation) method allows you to assemble the SOAP call on your own

January 12, 2010

ISE 431: Distributed Information Systems

:22

Creating a Java Web Service

- A compiler called wscompile is used to generate your WSDL file and stubs
 - You need to start with a configuration file that says something about the service you’re building and the interfaces that you’re converting into Web Services

January 12, 2010

ISE 431: Distributed Information Systems

:23

Example Configuration File

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
  <service name="StockQuote"
    targetNamespace="http://example.com/stockquote.wsdl"
    typeNamespace="http://example.com/stockquote/types"
    packageName="stockqt">

    <interface name="stockqt.StockQuoteProvider"
      servantName="stockqt.StockQuoteServiceImpl"/>

  </service>
</configuration>
```

January 12, 2010

ISE 431: Distributed Information Systems

:24

Starting a WAR

- The Web Service version of a Java JAR file is a Web Archive, WAR
- There's a tool called wsdeploy that generates WAR files
- Generally this will automatically be called from a build tool such as Ant
- Finally, you may need to add the WAR file to the appropriate location in Apache Tomcat (or WebSphere, etc.) and enable it
- See <http://java.sun.com/developer/technicalArticles/WebServices/WSPack2/jaxrpc.html> for a detailed example

January 12, 2010

ISE 431: Distributed Information Systems

:25

Finding a Web Service

- UDDI: Universal Description, Discovery, and Integration registry
- Think of it as DNS for web services
 - It's a replicated database, hosted by IBM, HP, SAP, MS
- UDDI takes SOAP requests to add and query web service interface data

January 12, 2010

ISE 431: Distributed Information Systems

:26

What's in UDDI

- White pages:
- Information about business names, contact info, Web site name, etc.
- Yellow pages:
- Types of businesses, locations, products
 - Includes predefined taxonomies for location, industry, etc.
- Green pages – what we probably care the most about:
- How to interact with business services; business process definitions; etc
 - Pointer to WSDL file(s)
 - Unique ID for each service

January 12, 2010

ISE 431: Distributed Information Systems

:27

Data Types in UDDI

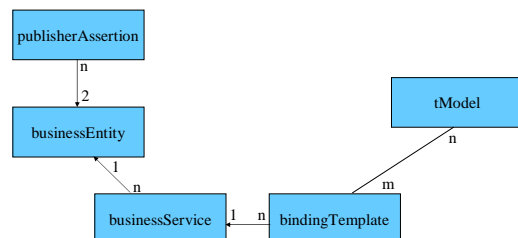
- **businessEntity**: top-level structure describing info about the business
- **businessService**: name and description of a service
- **bindingTemplate**: how to access the service
- **tModel** (t = type/technical): unique identifier for each service-template specification
- **publisherAssertion**: describes relationship between **businessEntities** (e.g., department, division)

January 12, 2010

ISE 431: Distributed Information Systems

:28

Relationships between UDDI Structures



January 12, 2010

ISE 431: Distributed Information Systems

:29

Example UDDI businessEntity

```

<businessEntity businessKey="0123..." xmlns="urn:uddi-org:api_v2">
  <discoveryURLs>
    <discoveryURL useType="businessEntity">
      http://uddi.ibm.com/registry/uddiget?businessKey=0123...
    </discoveryURL>
  </discoveryURLs>
  <name>My Books</name>
  <description>Technical Book Wholesaler</description>
  ...
  <businessServices>
    ...
  </businessServices>
  <identifierBag>
    <!-- keyedReferences to tModels -->
  </identifierBag>
  <categoryBag> ... </categoryBag>
</businessEntity>
  
```

January 12, 2010

ISE 431: Distributed Information Systems

:30

UDDI in Perspective

- Original idea was that it would just organize itself in a way that people could find anything they wanted
- Today UDDI is basically a very simple catalog of services, which can be queried with standard APIs
 - It's not clear that it really does what people really want: they want to find services "like Y" or "that do Z"

The Problem: With UDDI and Plenty of Other Situations

There's no universal, unambiguous way of describing "what I mean"

- Relational database idea of "normalization" doesn't convert concepts into some normal form – it just helps us cluster our concepts in meaningful ways
- "Knowledge representation" tries to encode definitions clearly – but even then, much is up to interpretation

The best we can do: describe *how things relate*

- *pollo* = *chicken* = *poulet* = 雞 = 鸡 = *jī* = *मुर्गी* = *murg*
- Note that this *mapping* may be imprecise or situation-specific!
 - Calling someone a chicken, vs. a chicken that's a bird