Web Services

CS 640, Lecture 14

Lecture outline

- Web Services
- AJAX
- Presentation layer for RPC

What do web services do?

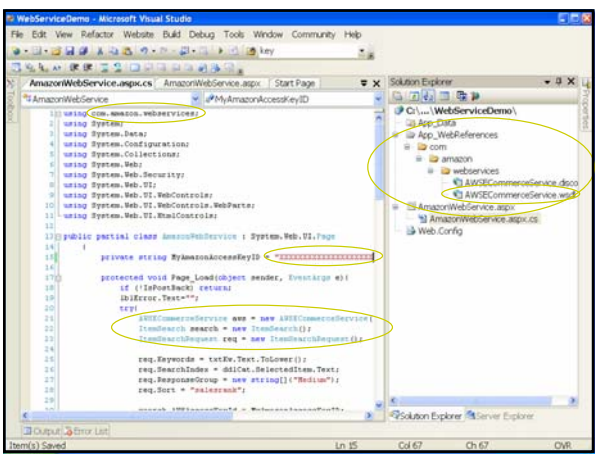
- Main use: ask another computer to run a procedure for you – parameters and results sent over network
- Remote Procedure Calls (RPC) -- a very old and very powerful idea
- Distinguishing features of web services
  - They run over http
  - They use XML to encode responses (and requests)
  - Client and server often use different languages
  - Client and server are often in different organizations
  - Client may be JavaScript code in browser – AJAX

More on web services

- Many companies allow access to their data / services through web services (some free)
  - Examples: Amazon, ebay, Google, Yahoo, USPS
- Two popular flavors
  - SOAP – requests are in XML
  - REST – request format same as query strings of HTML forms (sequence of name-value pairs)
- WSDL (web service description language) file used to describe format of messages supported by a given web service

ASP.NET and web services

- Writing web services
  - Create new web site of type “web service”
  - Write C# methods exposed to clients
  - The system does the rest (incl. generating WSDL)
- Using web services
  - Add “web reference” to your site/project
    - Give URL of WSDL file
  - Use web service in your code
    - System adds “glue” – class definitions, code for building requests and parsing responses, etc.



Lecture outline

- Web Services
- AJAX
- Presentation layer for RPC

What is AJAX?

- Dynamic, interactive web pages that feel more like a desktop application than a static page
- Key technologies – Asynchronous requests, JavaScript, DOM manipulation, CSS (especially positioning of elements), web services using XML
- Core tenet – in response to user actions, do not refresh full page from server, send small requests as needed and update individual elements of web page
- Examples: dragging/zooming maps at Google and Yahoo, “tooltip details” at Netflix, etc.

The XMLHttpRequest object

LiveSearch.js

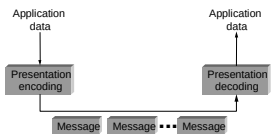
```
var xmlHttp;
function createXMLHttpRequest() {
  if (window.ActiveXObject) {
    return new ActiveXObject("Microsoft.XMLHTTP");
  } else if (window.XMLHttpRequest) {
    return new XMLHttpRequest();
  }
}
function do_liveSearch(pattern) {
  xmlHttp = createXMLHttpRequest();
  xmlHttp.onreadystatechange = handleStateChange;
  xmlHttp.open("GET",document.URL.substring(0,document.URL.lastIndexOf("/"))+
                "/SearchService.aspx?pattern="+pattern, true);
  xmlHttp.send(null);
}
function handleStateChange() {
  if(xmlHttp.readyState == 4) {
    if(xmlHttp.status == 200) {
      updatePage();
    } else {alert("Status "+xmlHttp.status);}
  }
}
... // var xmlDoc = xmlHttp.responseXML;
```

Lecture outline

- Web Services
- AJAX
- Presentation layer for RPC

Presentation Formatting

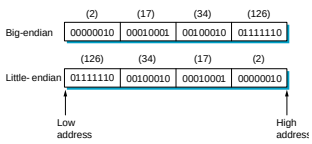
- Marshalling (encoding) application data into messages
- Unmarshalling (decoding) messages into application data
- Data types we consider
  - integers
  - floats
  - strings
  - arrays
  - structs



- Types of data we do not consider
  - images
  - video
  - multimedia documents

Difficulties

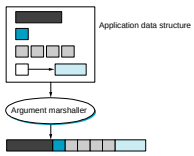
- Representation of base types
  - floating point: IEEE 754 versus non-standard
  - integer: big-endian versus little-endian (e.g., 34,677,374)



- Compiler layout of structures

Taxonomy

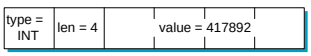
- Data types
  - base types (e.g., ints, floats); must convert
  - flat types (e.g., structures, arrays); must pack
  - complex types (e.g., pointers); must linearize



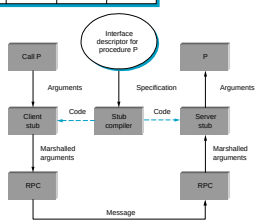
- Conversion Strategy
  - canonical intermediate form
  - receiver-makes-right (an *N* x *N* solution)

Taxonomy (cont)

- Tagged versus untagged data



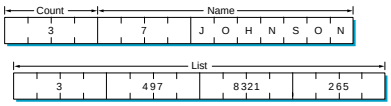
- Stubs
  - compiled
  - interpreted



eXternal Data Representation (XDR)

- Defined by Sun for use with SunRPC
- C type system (without function pointers)
- Canonical intermediate form
- Untagged (except array length)
- Compiled stubs

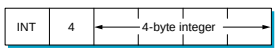
```
#define MAXNAME 256;
#define MAXLIST 100;

struct item {
   int     count;
   char    name[MAXNAME];
   int     list[MAXLIST];
};

bool_t
xdr_item(XDR *xdrs, struct item *ptr)
{
    return(xdr_int(xdrs, &ptr->count) &&
        xdr_string(xdrs, &ptr->name, MAXNAME) &&
        xdr_array(xdrs, &ptr->list, &ptr->count,
                MAXLIST, sizeof(int), xdr_int));
}
```

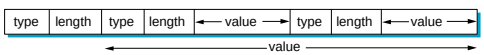


Abstract Syntax Notation One (ASN-1)

- An ISO standard
- Essentially the C type system
- Canonical intermediate form
- Tagged
- Compiled or interpretted stubs
- BER: Basic Encoding Rules

(tag, length, value)





Network Data Representation (NDR)

- Defined by DCE (CORBA)
- Basically the C type system
- Receiver-makes-right (architecture tag)
- Individual data items untagged
- Compiled stubs from IDL
- 4-byte architecture tag

- IntegerRep
  - 0 = big-endian
  - 1 = little-endian
- CharRep
  - 0 = ASCII
  - 1 = EBCDIC
- FloatRep
  - 0 = IEEE 754
  - 1 = VAX
  - 2 = Cray
  - 3 = IBM