CS 580 Client-Server Programming
Spring Semester, 2009
Doc 18 REST, Jar & Distributed Computing
23 Apr 2009

Copyright ©, All rights reserved. 2009 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/opl.shtml) license defines the copyright on this document.
References

REST Comment, http://developers.slashdot.org/article.pl?sid=03/04/03/1942235&mode=nocomment[tid=185&tid=156


Packaging Programs in JAR Files, http://java.sun.com/docs/books/tutorial/deployment/jar/


JSON - JavaScript Object Notation,
   Main JSON web site, http://www.json.org/

Java's Object Streams
   Java 6 API documentation, http://java.sun.com/javase/6/docs/api/
tadghin:
“I was recently talking with Jeff Barr, creator of syndic8 and now Amazon's chief web services evangelist. He let drop an interesting tidbit. Amazon has both SOAP and REST interfaces to their web services, and 85% of their usage is of the REST interface."

"Despite all of the corporate hype over the SOAP stack, this is pretty compelling evidence that developers like the simpler REST approach."
History

Roy Fielding

2000 Ph.D. Thesis

Architectural Styles and the Design of Network-based Software Architectures

What makes the Web scale?
REST Principles

Application state and functionality are abstracted into resources

Every resource is uniquely addressable using a universal syntax for use in hypermedia links

All resources share a uniform interface for the transfer of state between client and resource, consisting of

- A constrained set of well-defined operations
- A constrained set of content types, optionally supporting code on demand

A protocol which is:
- Client-server
- Stateless
- Cacheable
- Layered
Two Meanings of REST

1

Fielding's definition

2

Any simple interface which transmits domain-specific data over HTTP
Jar Files

Zip compressed files

Optional MANIFEST file

http://java.sun.com/docs/books/tutorial/deployment/jar/
Example

File: sdsu/Foo.class
Showing source rather than byte code

package sdsu;

public class Foo {

    public static void main(String[] args) {
        System.out.println(hello());
    }

    public static String hello() {
        return "Hello World";
    }

}
Creating the Jar file

> jar -cmf sampleManifest sdsu

Running the jar file

> java -jar sample.jar
Extracting contents of Jar

> jar -xvf sample.jar

created: META-INF/
inflated: META-INF/MANIFEST.MF
created: sdsu/
inflated: sdsu/Foo.class

Note that the MANIFEST.IM always ends up in META-INF
META-INF Directory

MANIFEST.MF
    Defines extension and package related data

INDEX.LIST
    Used by class loaders to speed up class loading

x.SF
    Signature file

x.DSA
    Signature of the corresponding signature file

services/
    Service provider configuration files
Some MANIFEST Attributes

Manifest-Version:
Created-By:
Signature-Version:
Class-Path:
Main-Class:

http://java.sun.com/javase/6/docs/technotes/guides/jar/jar.html
File: cs580/Bar.java

package cs580;

import sdsu.Foo;

public class Bar {

    public static void main(String[] args) {
        System.out.println("Here");
        System.out.println(Foo.hello());
    }

}
Adding Jar
Now Create Jar from Project

1. File
2. Select
3. Runnable JAR File Export
   - Select an export destination:
     - type filter text
     - JAR file
     - Javadoc
     - Runnable JAR file
4. Runnable JAR File Export
   - This operation repacks referenced libraries.
   - Please review the licenses associated with libraries you wish to reference to make sure you are able to repack them using this application. Note also that this operation does not copy signature files from original libraries to the generated JAR file.
   - Do not show this message again
   - Cancel
   - OK
New Jar File works

> 7java -jar test.jar
Here
Hello World

But No Jar inside the jar

> jar -tf test.jar
META-INF/MANIFEST.MF
cs580/Bar.class
sdsu/Foo.class
Distributed Computing
Jar's In Jars

You need to implement a class loader to load a jar file that is inside a jar file
Related Terms

Concurrent Computing
Simultaneous execution of multiple interacting computational tasks

Networking
Multiple computers interacting via network
Do not share single program

Distributed Computing
Different parts of a program run on multiple computers
Parts communicate via network

Parallel computing
Different parts of a program run on multiple processors in same computer
Some Motivation
Basic Communication Steps

Design protocol

Create domain objects

Extract protocol string from select domain objects

Convert protocol string to domain object
public class Vote {
    static String CR = "\r";
    int id;
    String option;

    public Vote(int pollId, String optionVote) {
        id = pollId;
        option = optionVote;
    }

    public int id() {return id;}

    public String option() {return option;}

    public String toString() {
        StringBuffer protocol = new StringBuffer();
        protocol.append("command:VOTE");
        protocol.append(CR);
        protocol.append("poll-id:");
        protocol.append(id);
        protocol.append(CR);
        protocol.append("option:");
        protocol.append(option);
        protocol.append(CR);
        return protocol.toString();
    }
}

public static Vote fromString(String voteString) {
    String[] lines = voteString.split(CR);
    HashMap<String, String> data = new HashMap<String, String>();
    for (int k = 0; k < lines.length; k++) {
        String[] keyValue = lines[k].split(":");
        data.put(keyValue[0].toLowerCase(), keyValue[1]);
    }
    return fromMap(data);
}

public static Vote fromMap(Map<String, String> voteData) {
    String option = voteData.get("option");
    Integer id = Integer.valueOf(voteData.get("poll-id"));
    return new Vote(id.intValue(), option);
}
Repeat

Repeat for each Command

Repeat for each client-server project
Some Ways to Automate the Work

JSON
ObjectStreams
JSON

http://www.json.org/

JavaScript Object Notation

data-interchange format

rfc 4627

Maps to/from strings
  null
  true, false
  number
  string
  array
  objects

Implementations in
  C, C++, C#, D, E, Java, Objective C
  Cold Fusion, Delphi, Erlang, Haskell
  JavaScript, Lisp, LotusScript, Perl,
  PHP, Pike, Prolog, Python, Ruby, Smalltalk
JSON Definition

Source: http://www.json.org/
String

Any UNICODE character except " or \ or control character

- quotation mark
- reverse solidus
- solidus
- backspace
- formfeed
- newline
- carriage return
- horizontal tab
- 4 hexadecimal digits
Number

number

0

digit

digit 1-9

digit

E

e

+

digit

-
## Examples

<table>
<thead>
<tr>
<th>Java Structure</th>
<th>JSON Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector array = new Vector();</td>
<td>[12,&quot;Egypt&quot;,false,-31]</td>
</tr>
<tr>
<td>array.append(new Integer(12));</td>
<td></td>
</tr>
<tr>
<td>array.append(&quot;Egypt&quot;);</td>
<td></td>
</tr>
<tr>
<td>array.append(new Boolean(false));</td>
<td></td>
</tr>
<tr>
<td>array.append(new Integer(-31));</td>
<td></td>
</tr>
</tbody>
</table>

```java
HashMap<String,Integer> object = new HashMap<String,Integer>();
object.put("lowerBound", 18);
object.put("upperBound", 139);
```

```json
{"lowerBound":18,"upperBound":139}
```
Possible Client-Server Usage

Use JSON as protocol syntax

Use JSON libraries to

Generate client-server messages

Parse messages from network
JSON in Java

http://www.json.org/java/index.html

A Java JSON library

JSONObject
  Constructs JSON strings
  Parses JSON strings

JSONObject json = new JSONObject();
json.put("lowerBound", 18);
json.put("upperBound", 139);

String objectString = json.toString(); //{"lowerBound":18,"upperBound":139}"

JSONObject newJson = new JSONObject(objectString);
int bound = newJson.getInt("lowerBound");  // 18
import org.json.JSONObject;
import org.json.JSONException;

public class Vote implements Serializable {
    int id;
    String option;

    public Vote(int pollId, String optionVote) {
        id = pollId;
        option = optionVote;
    }

    public int id() {return id;}
    public String option() {return option;}

    public String toJson() throws JSONException {
        JSONObject json = new JSONObject();
        json.put("command", "VOTE");
        json.put("poll-id", id);
        json.put("option", option);
        return json.toString();
    }

    public static Vote fromJson(String jsonString) throws JSONException {
        JSONObject json = new JSONObject(jsonString);
        int id = json.getInt("poll-id");
        String option = json.getString("option");
        return new Vote(id, option);
    }
}

Using Java JSON library from
http://www.json.org/java/index.html
Using JSON Strings

Vote cat = new Vote(1,"cat{:}dog");
String json = cat.toJson();

System.out.println(json);  //{"command":"VOTE","option":"cat{:}dog","poll-id":1}

Vote jsonVote = Vote.fromJson(json);
assertEquals(jsonVote.id(), 1);
Consequences

Benefits

Parsing and generation of protocol simplified
  No need to escape special characters

Define protocol in terms of
  maps (key-value pairs)
  arrays
  basic types (string, number, boolean)

Cross language support

Drawbacks

Nested { } and [] complicate parsing
No general end of message sequence
Limited support for primitive types
Object Streams

ObjectOutputStream
  Serializes objects
  Converts objects to bytes

ObjectInputStream
  Deserializes objects
  Converts DataOutputStream byte back into objects

"Requires" writer and reader to be Java programs
Possible Client-Server Usage

Create Message and Response classes

Send message objects to server

Message object

  Contains needed data

  Possibility executes methods on server
import java.io.Serializable;

public class Vote implements Serializable, Message {
    int id;
    String option;

    public Vote(int pollId, String optionVote) {
        id = pollId;
        option = optionVote;
    }

    public int id() {return id;}

    public String option() {return option;}

    public Response execute(VoteServer aServer) {
        boolean succeeded = aServer.addVote(id, option);
        if (succeeded)
            return new SuccessResponse();
        return new FailedResponse();
    }
}
Using ObjectStreams

Writing the Object
Vote cat = new Vote(1, "cat");
FileOutputStream catBytes = new FileOutputStream("cat");
ObjectOutputStream out = new ObjectOutputStream(catBytes);
out.writeObject(cat);
out.close();

Reading the Object
ObjectInputStream in = new ObjectInputStream(new FileInputStream("cat"));
Vote result = (Vote) in.readObject();

Output File
"ÍsrVoteãI€ä…álidLoptiontLjava/lang/String;xptcat
Sample Client Usage

Socket connection = new Socket(server, port);
OutputStream rawOut = connection.getOutputStream();
ObjectOutputStream out = new ObjectOutputStream(rawOut);
InputStream rawIn = connection.getInputStream();
ObjectInputStream in = new ObjectInputStream(rawIn);

Vote forCat = new Vote(1,"cat");
out.writeObject(forCat);
out.flush();
Response answer =(Response) in.readObject();

out.close();
in.close();
Sample Server

ServerSocket input = new ServerSocket( port );
while (true) {
    Socket client = input.accept();

    OutputStream rawOut = client.getOutputStream();
    ObjectOutputStream out = new ObjectOutputStream(rawOut);
    InputStream rawIn = client.getInputStream();
    ObjectInputStream in = new ObjectInputStream(rawIn);

    Message request = (Message) in.readObject();
    Response answer = request.execute(this);
    out.writeObject(answer);
    out.flush();
    client.close();
}
Consequences

Benefits

No need for a text protocol - just send objects

Protocol is just objects one can send

No need for parsing - just read objects

Drawbacks

Client and server need to be in Java

Client & server must have same classes

Modifications of a class can cause problems
Circular References

Care is needed when serializing/deserializing objects with circular references

ObjectStream does it correctly
Object Stream Protocol

The protocol used by Object Stream is documented at:

http://java.sun.com/javase/6/docs/platform/serialization/spec/protocol.html

At least one Lisp implementation of this protocol exists
RPC - Remote Procedure Call

Client "directly" calls a function on the server

Issues

Cross platform

Marshalling/unmarshalling of parameters and results
   How can one handle pointers as parameters?

Different contexts of client and server

Registering and finding servers
XML-RPC

RPC using
   HTTP as transport layer and
   XML to encode request/response
   Language and platform independent

Started by Userland (http://frontier.userland.com/) in 1998

Languages/Systems with XML-RPC implementations

   Java, Perl, Python, Tcl, C, C++, Smalltalk
   ASP, PHP, AppleScript, COM
   Zope, WebCrossing

Led to the development of SOAP
Example - Add Server

```java
import org.apache.xmlrpc.*;

public class AddServer {
    public Integer addtwo(int x, int y) {
        return new Integer(x + y);
    }

    public static void main( String[] args) {
        try {
            System.out.println("Starting server on port 8080");
            WebServer addTwoServer = new WebServer(8080);
            addTwoServer.addHandler("examples", new AddServer());
            addTwoServer.start();
            System.out.println("server running");
        }
        catch (Exception webServerStartError) {
            System.err.println("JavaServer " + webServerStartError.toString());
        }
    }
}
```

Client can access all public instance methods in AddServer.
Example - Client

```java
import java.util.*;
import org.apache.xmlrpc.*;

public class XmlRpcExample {
    public static void main(String args[]) {
        try {
            XmlRpcClient xmlrpc = new XmlRpcClientLite("http://127.0.0.1:8080/");
            Vector parameters = new Vector();
            parameters.addElement(new Integer(5));
            parameters.addElement(new Integer(3));

            Integer sum = (Integer) xmlrpc.execute("examples.addtwo", parameters);

            System.out.println(sum.intValue());
        } catch (java.net.MalformedURLException badAddress) {
            badAddress.printStackTrace(System.out);
        } catch (java.io.IOException connectionProblem) {
            connectionProblem.printStackTrace(System.out);
        } catch (Exception serverProblem) {
            serverProblem.printStackTrace(System.out);
        }
    }
}
```
Consequences

Benefits

Protocol = public methods
Handles the network communications
Handles generation/parsing of messages
Multiple language support
Platform independent
Simple

Drawbacks

Long messages
Limited support for objects