

Web Service XML RPC

1

AMRO NAJJAR
TELECOM SAINT-ETIENNE
MASTER WEB-INTELLIGENCE



References

2

- **Web Services: Principles and Technology**, Michael Papazoglou
- **Java Web Services: Up and Running**, Martin Kalin
- **Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTFUL web services**, Robert Daigneau
- **Web Services Essentials (O'Reilly XML)**, Ethan Cerami

XML-RPC

3

- provides an XML- and HTTP-based mechanism for making method or function calls across a network
- Very simple & useful
- This slides present:
 - An introduction to the main concepts and history of XML-RPC
 - XML-RPC usage scenarios

XML-RPC Intro

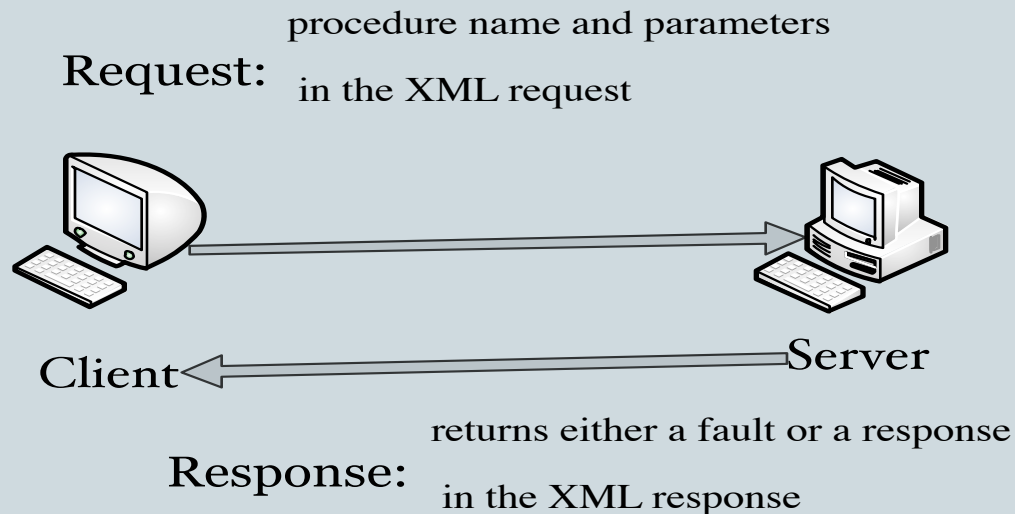
4

- Developed by Dave Winer of Userland in the late 1990
- Is a very lightweight RPC system
 - Support for elementary data types (basically, the built-in C types together with a boolean and a datetime type)
 - Few simple commands
 - Follows the request/response pattern
- Two key features
 - use of XML marshaling/unmarshaling to achieve language neutrality
 - Reliance on HTTP for transport

XML-RPC Overview

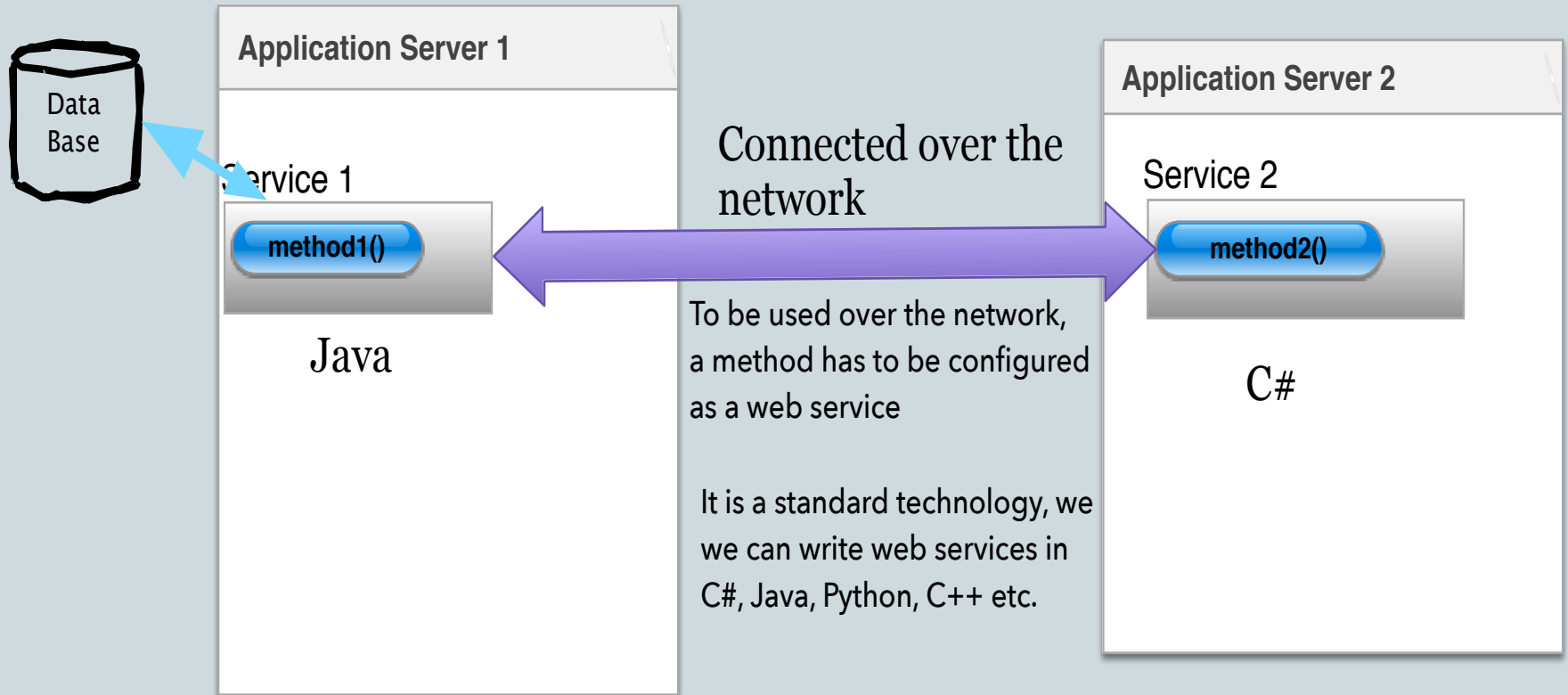
5

- Permits programs to make function or procedure calls across a network
- Use HTTP Protocol to pass information from a client computer to a server
 - Nature of requests and responses with a small XML vocabulary



XML RPC

6



XML RPC : Parameters

7

- XML-RPC parameters are a simple list of types and content
 - structs and arrays are the most complex types available.
- XML-RPC has no notion of objects
- No mechanism for including information that uses other XML vocabularies

Remark

This is a limitation to be addressed by the SOAP and co

XML RPC Use Scenarios

8

- **Glue Code with XML-RPC**
- **Publishing Services with XML-RPC**

Glue Code with XML-RPC

9

- Developers need to address the problem of integrating distributed systems
 - Unix systems need to speak with Windows etc.
- Instead of creating custom systems, **developers can use XML-RPC to connect programs running on different systems and environments**
- Use existing APIs and add connections to those APIs as necessary
- **No need for a clear distinction between Client & Server**

Publishing Services with XML-RPC

10

- XML-RPC can be used to publish information to the world
- It allows information recipients to be any kind of client that understands the XML-RPC interface

XML-RPC Technical Overview

11

- XML-RPC consists of three relatively small parts
 - *XML-RPC data model*
 - ✦ A set of types for use in passing parameters, return values, and faults (error messages)
 - *XML-RPC request structures*
 - ✦ An HTTP POST request containing method and parameter information
 - *XML-RPC response structures*
 - ✦ An HTTP response that contains return values or fault information

XML RPC is the combination of the three:

The data structures are used by both the request and response structures

XML-RPC Data Model: Basic Types

12

- six basic data types and two compound data types
 - this is a restricted set of types
 - Attempts to hit the lowest common denominator for many kinds of program-to- program communications
- Basic types are represented by simple XML elements
 - `<string>Hello World!</string>`
 - Basic types are always enclosed in value elements
 - ✦ `<value><int>7</int></value>`

XML-RPC Data Model: Basic Types Overview

13

```
<!-- ... string as ... -->
<value><string>an </string></value>
<value><string>array.</string></value>
</data>
</array>
</value>
```

The following `array` contains four integers:

```
<value>
  <array>
    <data>
      <value><int>7</int></value>
      <value><int>1247</int></value>
      <value><int>-91</int></value>
      <value><int>42</int></value>
    </data>
  </array>
</value>
```

XML-RPC Data Model: Arrays

14

- Array Example:

```
<value><int>-91</int>  
<value><int>42</int>  
</data>  
</array>  
</value>
```

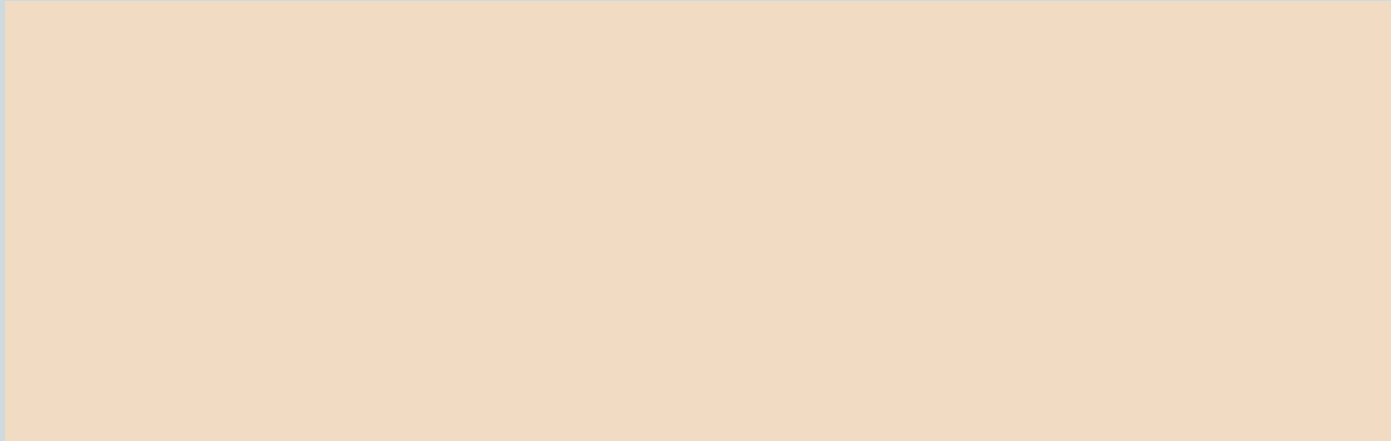
Remark

This array is composed of only one type (integer)

XML-RPC Data Model: Arrays & Struct

15

- Array Example: mixed array



- Structs:
 - contain unordered content, identified by name (name is string)
 - Each struct element contains a list of member elements
 - Member elements each contain one name element and one value element

XML-RPC Data Model: Struct Example

16

- A Struct Example:

kind of complex transfer.

- Structs can contain other structs, or even arrays
- Arrays can also contain structs

XML-RPC Request Structure

17

- Requests are a combination of XML content and HTTP headers.
- The XML content
 - uses the data typing structure to pass parameters
 - Contains information identifying the procedure being called
- HTTP headers provide a wrapper for passing the request over the Web
- A request contains a single XML document,
 - whose root element is a methodCall
 - Each methodCall element contains a methodName element and a params element

XML-RPC Request Example

18

- CircleArea , takes a Double parameter (for the radius), the XML-RPC request would look like

```
<param>  
  <value><double>18.2466842913</double></value>  
</param>  
</params>  
</methodResponse>
```

HTTP
Header

XML-RPC
Request

XML-RPC Response Structure

19

- much like requests, but we distinguish two possible cases:
 - If the response is successful :
 - ✦ the procedure was found, executed correctly, and returned results
 - If there was a problem in processing the XML-RPC request
 - ✦ The result will contain a fault message

XML-RPC Response : Success

20

- The response will look like the request
 - The methodCall element is replaced by a methodResponse element
 - There is no methodName element

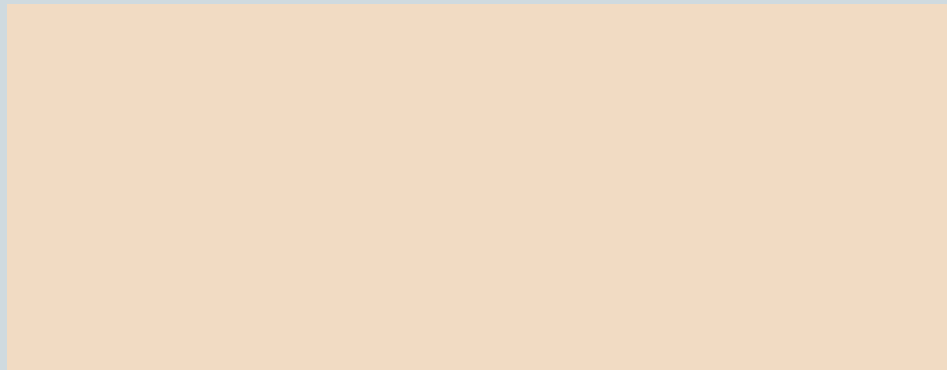
Remark

An XML-RPC response can only contain one parameter (this parameter can be an array or struct)

XML-RPC Response :Failure

21

- The methodResponse element will contain a fault element instead of a params element
- The fault element, like the params element, has only a single value
- that value indicates that something went wrong
- Fault example:



XML RPC : Pros & Cons

22

- limitations

- In the world of OOP, XML RPC seems too simple
- Complex types are not represented
 - ✦ You can, in some cases, use these complex types to represent object structures, but at some point you may find it easier to use SOAP for that kind of complex transfer.
- Does not exploit HTTP well
 - ✦ All XML-RPC responses use the 200 OK response code, even if a fault is contained in the message
- XML-RPC won't do anything to guarantee that arrays have a consistent number or type of values. You'll need to make sure that you write code that consistently generates the right number and type of output values if consistency is necessary for your application.

XML-RPC vs SOAP

23

XML RPC

- Very simple and limited (designed to be efficient)
- Short specification (originally 6 pages)
- Order of passed parameters is relevant
- More Python Support

SOAP

- Has a complete relevant ecosystem
- Long specification (about 50 pages)
- The order of passed parameters is irrelevant
- More powerful in general

Optional: XML RPC Cod Example

24

- In the circle example, we need the following simple method:

```
public class AreaServer {  
  
    public static void main(String[]
```


Java XML Server Example

25

```
public class AreaServer {  
    public static void main(String[]
```

**Needed
Libraries**

Now accepting requests. (Halt program to stop.)

The `AreaClient` class shown in Example 2-3 tests the `AreaServer`, on the command line. The `AreaClient` class also uses the XML-RPC library to use a few lines of code (in the `areaCircle` method) to make the actual call.

Main (): verifying
If the input
is pertinent
If yes: start the server

Example 2-3. A Java client to test the XML-RPC server

```
package com.ecerami.xmlrpc;  
  
import java.io.IOException;  
import java.util.Vector;  
import org.apache.xmlrpc.XmlRpc;  
import org.apache.xmlrpc.XmlRpcClient;  
import org.apache.xmlrpc.XmlRpcException;  
  
public class AreaClient {  
  
    public static void main(String args[]) {  
        if (args.length < 1) {
```

**The method that
launches the server
the argument is the
port**

Java XML Server Example

26

```
import org.apache.xmlrpc.XmlRpcClient;  
  
public class AreaClient {  
  
    public static void main(String[] args)  
        if (args.length < 1) {
```

**Needed
Libraries**

classes (like the `Double` object for `double` primitives) around the argument. Once a `Vector` has been constructed, it is fed to the `execute` method of the `XmlRpcClient` class. The `execute` method is a procedure being called. In this case, the name of the method being called is `circleArea`, reflecting that the `AreaHandler` class was registered on the server with the name `circleArea`. That it contains a method called `circleArea`.

Verify the Input

When the `execute` method is called, the client makes an XML-RPC request to the server. The request calls the method identified by the name `circleArea` and the arguments specified in its constructor. The request calls the method identified by the name `circleArea` and the arguments specified in its constructor.

**Create the client
method : *areaCircle()***

The method : addCircle()

27

vector has been constructed, it is led to the execute method along with procedure being called. In this case, the name of the method is an reflecting that the `AreaHandler` class was registered on the server with th that it contains a method called `circleArea`.

When the `execute` method is called, the client makes an XML-RPC request specified in its constructor. The request calls the method identified by th

The line above is important

Create the client & Identify the server

Create the Parameters using The user's input

Issue a request with the Parameters and retrieve the results

The generated XML request

28

except that Perl's flexibility allows us to skip packaging. The program shown in Example 2-4 accepts a `radius` value from the command line, creates a new XML-RPC connection, and passes the `radius` value to the `area.circleArea` method. Then the program prints the re-

And the server responds with a `methodResponse`

XML RPC is technology-independent

We can create another client in C++ and another in Perl etc.