

LẬP TRÌNH PHÂN TÁN THEO CHỦ ĐỀ



XML-RPC

Cao Tuấn Dũng
Trịnh Tuấn Đạt

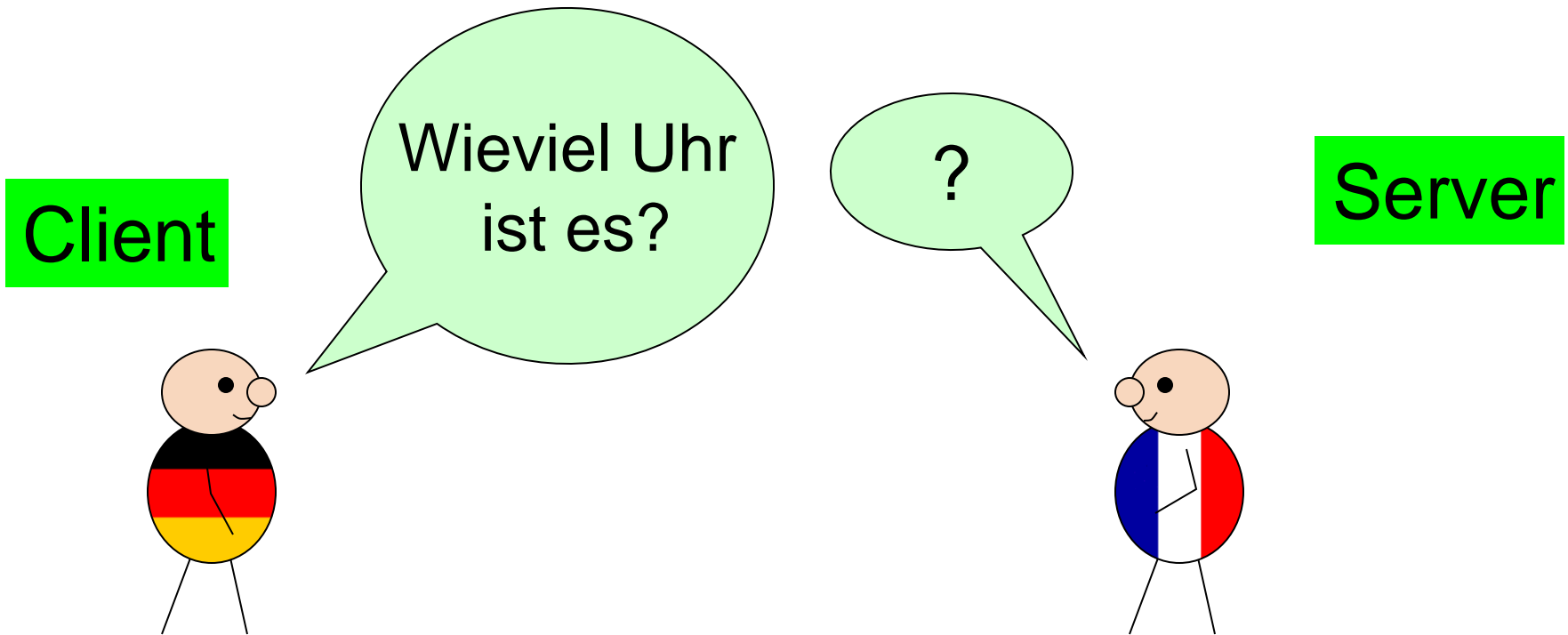


Nội dung

- 1. Giới thiệu
- 2. XML-RPC là gì
- 3. Chương trình ví dụ


1. Giới thiệu

- Client và Server cần hiểu nhau.



Ngôn ngữ chung thống nhất

Wieviel Uhr
ist es?

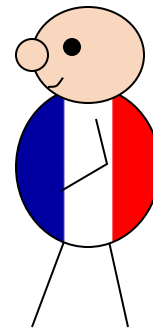
 What's the
time?

Quelle heure est-il?
7pm!

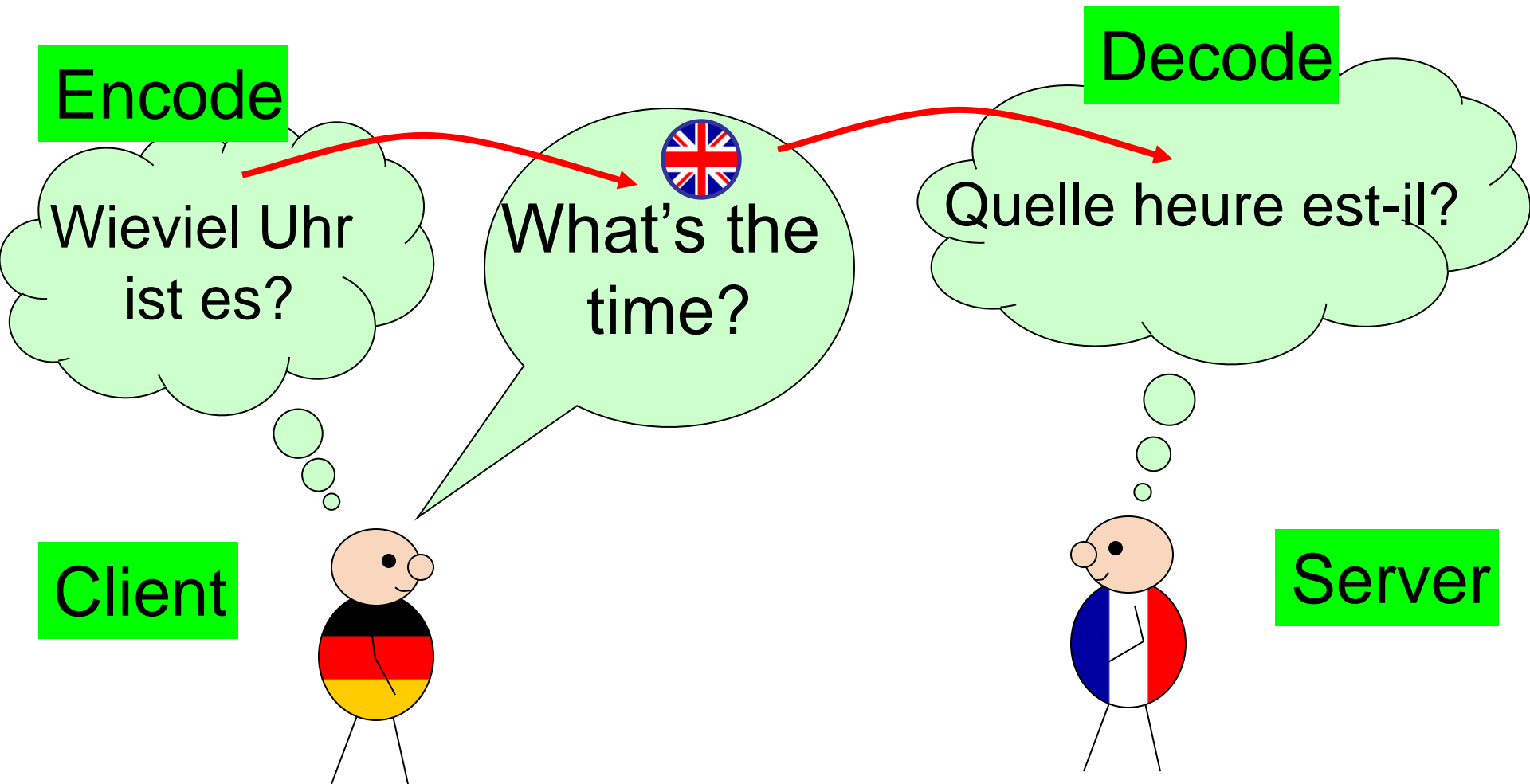
Client



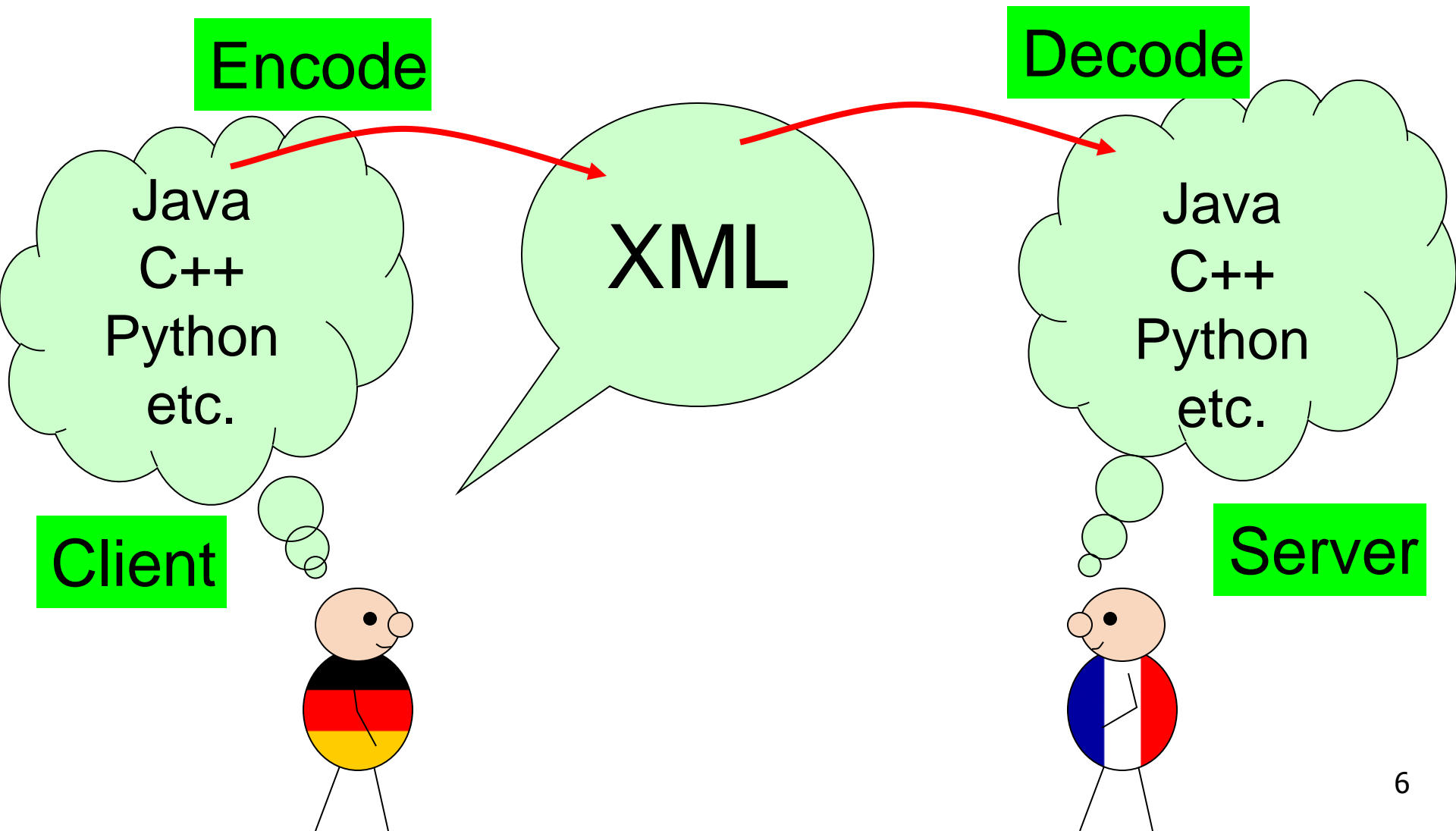
Server



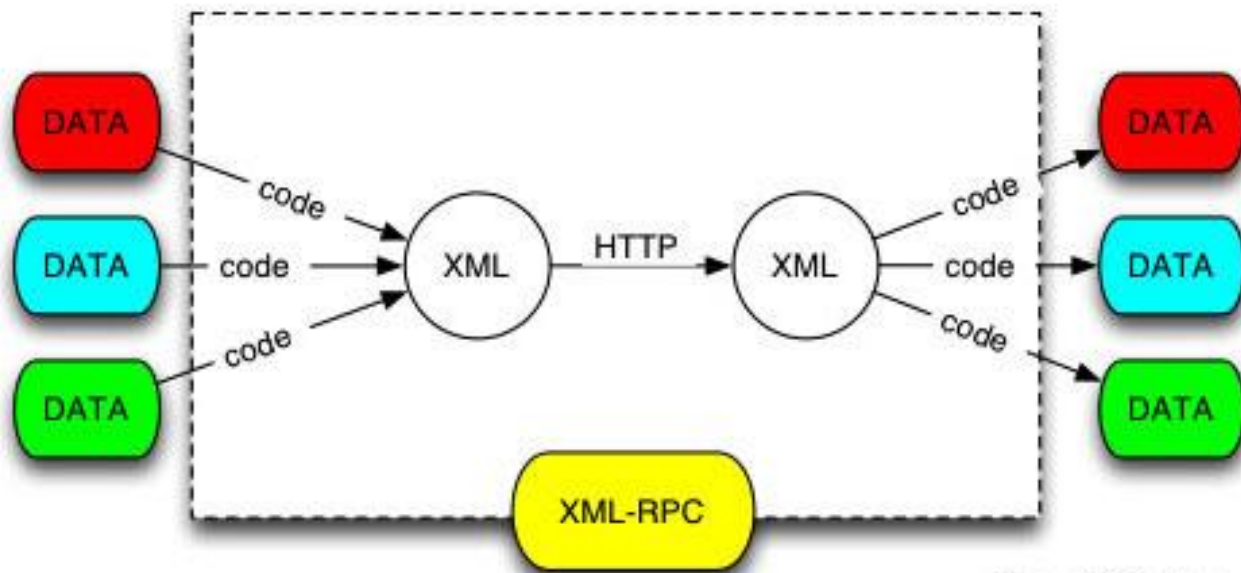
Client mã hóa thông tin dùng “ngôn ngữ chung”, server giải mã.



XML – ngôn ngữ truyền dữ liệu chung



XML - RPC



Source: JY Stervinou

Local procedure call

```
public class Example {  
    public int sum(int a, int b) {  
        return a+b;  
    }  
    public static void main (String [] args) {  
        Example eg = new Example();  
        eg.sum(13,17);  
    }  
}
```

Remote procedure call (RPC)

Client

```
[...]  
eg.sum(13,17);  
[...]
```



Server

```
[...]  
public int sum(int a, int b) {  
    return a+b;  
}  
[...]
```

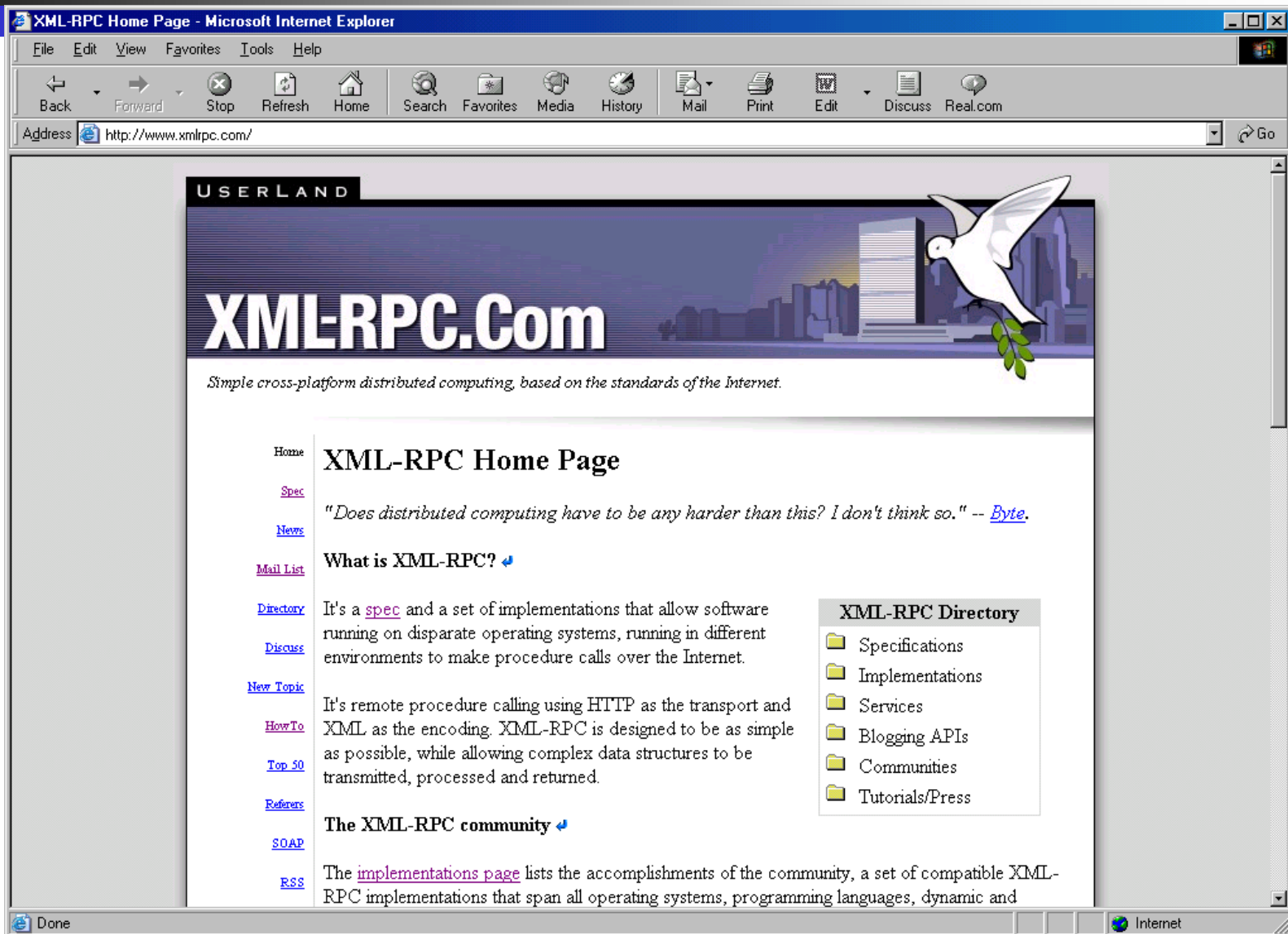


Nhắc lại về RPC

- RPC: kỹ thuật để xây dựng các ứng dụng phân tán, theo mô hình client-server.
- Là mở rộng của lời gọi thủ tục cục bộ thông thường.
- Như khái niệm “remote”, thủ tục được gọi không cần tồn tại ở cùng không gian địa chỉ với nơi gọi.
 - Hai tiến trình có thể ở cùng hệ thống, hoặc ở các hệ thống khác nhau, cùng kết nối vào 1 network.
- Nhờ sử dụng RPC, lập trình các ứng dụng phân tán sẽ tránh các bước chi tiết trong lập trình mạng.



2. XML-RPC: www.xmlrpc.com





XML-RPC là gì?

- “Đặc tả và tập các cài đặt” cho phép phần mềm chạy trên các hệ điều hành khác nhau, chạy trên các môi trường khác nhau, tạo ra các lời gọi thủ tục trên toàn mạng Internet
- Là lời gọi thủ tục từ xa sử dụng giao thức HTTP và XML để mã hóa. XML-RPC được thiết kế đơn giản nhất có thể, nhưng có thể truyền đi, xử lý, trả về các cấu trúc dữ liệu phức tạp.



Đặc tả lightweight.

- <1800 từ
- dễ học, có ví dụ

- “**Đặc tả** và tập các cài đặt” cho phép phần mềm chạy trên các hệ điều hành khác nhau, chạy trên các môi trường khác nhau, tạo ra các lời gọi thủ tục trên toàn mạng Internet
- Là lời gọi thủ tục từ xa sử dụng giao thức HTTP và XML để mã hóa. XML-RPC được thiết kế **đơn giản nhất có thể**, nhưng có thể truyền đi, xử lý, trả về các cấu trúc dữ liệu phức tạp.

- 
- “Đặc tả và **tập các cài đặt**” cho phép phần mềm chạy trên các hệ điều hành khác nhau,

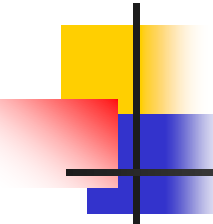
Các ngôn ngữ hỗ trợ:
C/C++, Java, Perl, Python,
Frontier, Lisp, PHP, Microsoft
.NET, Rebol, Real Basic, Tcl,
Delphi, WebObjects and Zope

, tạo ra
internet
o thức
được thiết
thể truyền
phức tạp.



Uses existing protocols (HTTP) and a well established framework (XML).

- “It's a spec and a set of implementations” cho phép phần mềm chạy trên các hệ điều hành khác nhau, chạy trên các môi trường khác nhau, tạo ra các lời gọi thủ tục trên toàn mạng Internet
- Là lời gọi thủ tục từ xa sử dụng giao thức **HTTP** và **XML** để mã hóa. XML-RPC được thiết kế đơn giản nhất có thể, nhưng có thể truyền đi, xử lý, trả về các cấu trúc dữ liệu phức tạp.



The following data structures are supported: integer, boolean, string, double, date & time, base64 binaries, structs, arrays.

phép phân mem chạy trên các hệ điều hành khác nhau, chạy trên các môi trường khác nhau, tạo ra các lời gọi thủ tục trên toàn mạng Internet

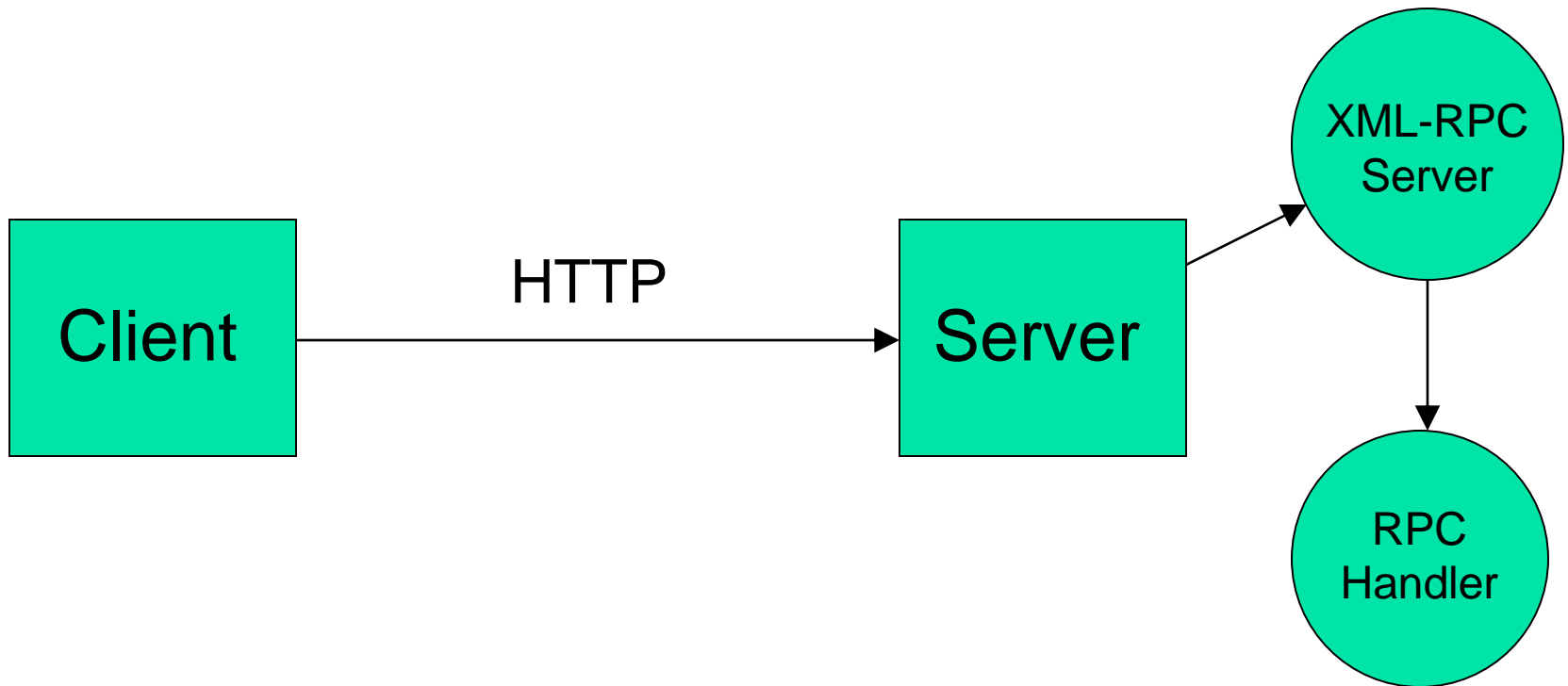
- Là lời gọi thủ tục từ xa sử dụng giao thức HTTP và XML để mã hóa. XML-RPC được thiết kế đơn giản nhất có thể, nhưng có thể truyền đi, xử lý, trả về các **cấu trúc dữ liệu phức tạp.**



Kiến trúc ứng dụng XML-RPC

- Server gồm ba thành phần:
 - main thread
 - Server XML-RPC (cài đặt các stub)
 - Một hoặc nhiều RPC handler
- Server xử lý mỗi RPC thông qua việc gọi các RPC handler tương ứng.

Kiến trúc ứng dụng XML-RPC





Kiểu dữ liệu

- Số nguyên có dấu 4-byte Boolean
- String
- Double
- Date/Time
- Binary (base-64)
- Mảng
- Cấu trúc



Thư viện

- C++: xmlrpc-c
 - <http://xmlrpc-c.sourceforge.net>
- Java: Apache XML-RPC
 - <http://ws.apache.org/xmlrpc>
 - Yêu cầu
 - <http://jakarta.apache.org/commons/codec>



Chế độ tương tác

- Hai kiểu
 - Đồng bộ
 - Không đồng bộ



Đồng bộ

- Client thực thi thủ tục trên server
 - Client bị khóa cho đến khi RPC trả về kết quả
- Server trả về kiểu (đối tượng) tổng quát
 - C++ (xmlrpc-c)
 - xmlrpc_c::value
 - Java (Apache)
 - Object
- Client phải biết mình cần kiểu dữ liệu gì
 - Ép kiểu



Không đồng bộ

- Client phải định nghĩa một đối tượng với
 - Phương thức *call-back* xử lý khi RPC thành công
 - Phương thức *call-back* xử lý khi RPC thất bại/lỗi
- Client
 - Sinh một thread cho RPC
 - Quay về
- Khi RPC kết thúc, một trong các phương thức *call-back* sẽ được gọi.
 - Ép kiểu giá trị trả về



Thực hiện XML-RPC

- 1. Lên danh sách tham số
- 2. Khởi tạo RPC
 - Tên
 - URL của server
 - `http://address:port/RPC1`
- 3. Thi hành RPC
- 4. Xử lý lỗi
- 5. Xử lý giá trị trả về



Ví dụ: một ứng dụng XML-RPC client/server viết bằng Java.

- Gói Java `org.apache.xmlrpc` cung cấp các Classes để cài đặt một XML-RPC client và một XML-RPC server.
 - Tìm trên: <http://ws.apache.org/xmlrpc/>
 - Thêm `cis69mc.jar` vào Classpath
- Thủ tục từ xa: thực hiện phép cộng



Lên danh sách tham số

- C++

```
xmlrpc_c::paramList params;  
params.add(  
    xmlrpc_c::value_string("Hello server."));
```

- Java

```
java.util.Vector params = new Vector();  
params.add(new String("Hello server."));
```



Khởi tạo RPC

■ C++

```
xmlrpc_c::clientXmlTransport_libwww xmlrpcTransport;  
xmlrpc_c::client_xml xmlrpcClient(&xmlrpcTransport);  
xmlrpc_c::rpcPtr xmlrpc("server.sayHello", params);  
xmlrpc_c::carriageParm_libwww0 cParm(  
    "http://127.0.0.1:9382/RPC2");
```



Khởi tạo RPC

■ Java

```
XmlRpcClient client = null;  
try {  
    client = new XmlRpcClient(  
        "http://127.0.0.1:9382/RPC2");  
} catch (MalformedURLException e) {  
    System.err.println(e);  
}
```



Thực thi (gọi) RPC

■ C++

```
try {  
    xmlrpc->call(&client, &cParm);  
} catch (std::exception const e) {  
    cerr << "Connection refused." << endl;  
}
```



Thực thi (gọi) RPC

■ Java

```
try {  
    Object returnValue = client.execute(  
        "server.sayHello", params);  
} catch (XmlRpcException e) {  
    System.err.println(e); // some RPC problem  
} catch (IOException e) {  
    System.err.println("Connection refused.");  
}
```



Xử lý lỗi

■ C++

```
if (xmlrpc->isSuccessful() == false) {  
    if (xmlrpc->isFinished() == true) {  
        xmlrpc_c::fault f = xmlrpc->getFault();  
        if (f.getCode() == 0) {...} // exception  
        else {...} // unexpected error  
    }  
    else {...} // unexpected error  
}
```



Xử lý kết quả

■ C++

```
if (xmlrpc->isSuccessful() == true) {  
    xmlrpc_c::value v = xmlrpc->getResult();  
    xmlrpc_c::value_string rpcstr =  
        (xmlrpc_c::value_string) v;  
    std::string text = (std::string) rpcstr;  
}
```



Xử lý kết quả

- Java

```
try {  
    String text = (String) returnValue;  
} catch (ClassCastException e) {  
    // returnValue was not a String.  
    // It could have been an Exception.  
}
```




Xây dựng Server

- 1. Định nghĩa một hay nhiều handlers
 - C++: Lớp con của `xmlrpc_c::method`
 - Java: tạo ra một lớp public mới
- 2. Khởi tạo đối tượng server (cung cấp bởi thư viện)
 - Chỉ định cổng
- 3. Thêm handlers vào server
- 4. Khởi động server
 - Có thể tạo ra thread mới



Định nghĩa các Handler

■ C++

```
class HelloMethod : public xmlrpc_c::method {  
    public:  
        void execute(xmlrpc_c::paramList  
            const& params, xmlrpc_c::value*  
            const retvalP) {  
            *retvalP = xmlrpc_c::value_string(  
                "Hello client.");  
        }  
};
```



Định nghĩa các Handler

■ Java

```
public class RPCHandler {  
    // All public methods in this class are  
    // RPC handlers  
    public String sayHello(String param) {  
        return new String("Hello client.");  
    }  
}
```



Khởi tạo Server

- C++

```
xmlrpc_c::registry reg;  
xmlrpc_c::serverAbyss server;
```

- Java

```
WebServer server = null;  
try {  
    server = new WebServer(9382) ;  
} catch (Exception e) {  
    System.err.println(e) ;  
}
```



Thêm Handler vào server

■ C++

```
xmlrpc_c::methodPtr const helloMethod(  
    new HelloMethod);  
reg.addMethod("server.sayHello",  
    helloMethod);  
server = xmlrpc_c::serverAbyss(reg, 9382,  
    "xmlrpc.log");
```



Thêm Handler

- Java

```
server.addHandler(  
    "server",  
    new RPCHandler()  
);
```



Khởi động server

- C++

// Chạy trong luồng hiện tại.
`server.run();`

- Java

// Chạy trong luồng mới.
`server.start();`

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

Java Client



```
public class JavaClient {  
    public static void main (String [] args) {  
        try {  
            XmlRpcClient server = new XmlRpcClient("http://localhost/RPC2");  
            Vector params = new Vector();  
            params.addElement(new Integer(17));  
            params.addElement(new Integer(13));  
            Object result = server.execute("sample.sum", params);  
            int sum = ((Integer) result).intValue();  
            System.out.println("The sum is: "+sum);  
        } catch (Exception exception) {  
            System.err.println("JavaClient: " + exception);  
        }  
    }  
}
```




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

```
public class JavaClient {  
    public static void main (String[] args) {  
        try {  
            XmlRpcClient server = new XmlRpcClient("http://localhost:8080/xmlrpc");  
            Vector params = new Vector();  
            params.addElement(new Integer(1));  
            params.addElement(new Integer(2));  
            Object result = server.execute("sum", params);  
            int sum = ((Integer) result).intValue();  
            System.out.println("The sum is: " + sum);  
        } catch (Exception exception) {  
            System.err.println("JavaClient: " + exception);  
        }  
    }  
}
```

■ Gói Java org.apache.xmlrpc chứa các lớp phục vụ cho XML-RPC Java clients và XML-RPC server. Ví dụ, cho client là XmlRpcClient.

■ Gói java.util sử dụng cho lớp Vector.



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

```
public class JavaClient {  
    public static void main (String [] args) {  
        try {  
            XmlRpcClient server = new XmlRpcClient("http://localhost/RPC2");  
            Vector params = new Vector();  
            params.addElement(new Integer(17));  
            params.addElement(new Integer(13));  
            Object result = server.execute("sample.sum", params);  
            int sum = ((Integer) result).intValue();  
            System.out.println("The sum is: "+sum);  
        } catch (Exception exception) {  
            System.err.println("JavaClient: " + exception);  
        }  
    }  
}
```



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

```
public class JavaClient {  
    public static void main (String [] args) {  
        try {  
            XmlRpcClient server = new XmlRpcClient("http://localhost/RPC2");  
            Vector params = new Vector();  
            params.addElement(new Integer(17));  
            params.addElement(new Integer(13));  
            Object result = server.execute("sample.sum", params);  
            // int sum = ((Integer) result).intValue();  
        }  
    }  
}
```

■ Dòng lệnh này sẽ gửi request đến server. Thủ tục `sum(17,13)` được gọi trên server như một thủ tục cục bộ. Giá trị trả về luôn là một Object.

■ “sample” là một *handler* được định nghĩa trên server.

```
import java.util.*;
```

Δ Java Client



- Các tham số của thủ tục gọi **luôn luôn** đặt trong Vector.

```
try {  
    XmlRpcClient server = new XmlRpcClient("http://localhost/RPC2");  
    Vector params = new Vector();  
    params.addElement(new Integer(17));  
    params.addElement(new Integer(13));  
    Object result = server.execute("sample.sum", params);  
}
```

- Dòng lệnh này sẽ gửi request đến server. Thủ tục sum(17,13) được gọi trên server như một thủ tục cục bộ. Giá trị trả về luôn là một Object.

- "sample" là một **handler** được định nghĩa trên server.



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

```
public class JavaClient {  
    public static void main (String [] args) {  
        try {  
            XmlRpcClient server = new XmlRpcClient("http://localhost/RPC2");  
            Vector params = new Vector();
```

- Đối tượng lớp XmlRpcClient được khởi tạo bằng cách chỉ ra "**web address**" của máy server.
 - localhost - chỉ ra máy cục bộ.
 - Địa chỉ IP, ví dụ: 194.80.215.219
 - Tên, ví dụ: cis69.dyndns.org
 - Trong tất cả các trường hợp trên, theo sau là một số hiệu cổng, ví dụ: cis69.dyndns.org:8080. mặc định là cổng 80.

- Vì kết quả của lời gọi thủ tục từ xa luôn là một Object, nên nó phải được ép về kiểu thích hợp. (Trường hợp này là Integer)



```
public static void main (String [] args) {  
    try {  
        XmlRpcClient server = new XmlRpcClient("http://localhost/RPC2");  
        Vector params = new Vector();  
        params.addElement(new Integer(17));  
        params.addElement(new Integer(13));  
        Object result = server.execute("sample.sum", params);  
        int sum = ((Integer) result).intValue();  
        System.out.println("The sum is: "+sum);  
    } catch (Exception exception) {  
        System.err.println("JavaClient: " + exception);  
    }  
}
```



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

```
public class JavaClient {  
    public static void main (String [] args) {  
        try {
```

```
            XmlRpcClient server = new XmlRpcClient("http://localhost:8080/xmlrpc");  
            Vector params = new Vector();  
            params.addElement(new Integer(1));  
            params.addElement(new Integer(2));  
            Object result = server.execute(params);  
            int sum = ((Integer) result).intValue();  
            System.out.println("The sum is: "+sum);
```

```
        } catch (Exception exception) {  
            System.err.println("JavaClient: " + exception);  
        }  
    }  
}
```

- Khi có lỗi(không có kết nối, ...), xử lý với try-catch.



Client



Server



```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<methodCall>
```

```
  <methodName>sample.sum</methodName>
```

```
  <params>
```

```
    <param>
```

```
      <value><int>17</int></value>
```

```
    </param>
```

```
    <param>
```

```
      <value><int>13</int></value>
```

```
    </param>
```

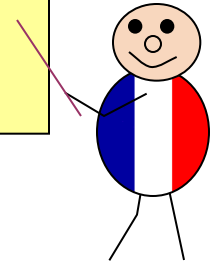
```
  </params>
```

```
</methodCall>
```

- Đây là những gì Client gửi cho server.


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

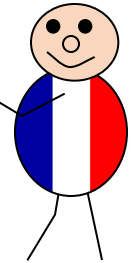
Java Server



```
public class JavaServer {  
    public Integer sum(int x, int y) {  
        return new Integer(x+y);  
    }  
    public static void main (String [] args) {  
        try {  
            WebServer server = new WebServer(80);  
            server.addHandler("sample", new JavaServer());  
            server.start();  
        } catch (Exception exception) {  
            System.err.println("JavaServer: " + exception);  
        }  
    }  
}
```

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

A Java Server



```
public class JavaServer {  
    public Integer sum(int x, int y) {  
        return new Integer(x+y);  
    }  
    public static void main (String [] args) {  
        try {  
            WebServer server = new WebServer(80);  
            server.addHandler("sample", new JavaServer());  
            server.start();  
        } catch (Exception exception) {  
            System.out.println(exception);  
        }  
    }  
}
```

- Gói org.apache.xmlrpc chứa lớp WebServer để cài đặt một XML-RPC Server

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

```
public class JavaServer {  
    public Integer sum(int x, int y) {  
        return new Integer(x+y);  
    }  
}
```

```
public static void main (String [] args) {  
    try {
```

```
        WebServer server = new WebServer(80);
```

```
        server.addHandler("sample", new JavaServer());
```

```
    } catch (Exception e) {
```

```
    }
```

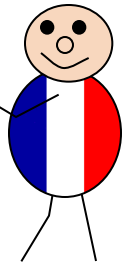
```
    System.out.println("Server is running on port 80");
```

```
}
```

```
}
```

```
}
```

A Java Server



- Thủ tục được gọi từ xa phải được cài đặt là 1 phương thức public trong 1 class

- Một thể hiện của lớp này được kết hợp với một handler truy cập được từ client.

```
import org.ap
```

```
public class J
```

```
public Integer
```

```
return new
```

```
}
```

```
public static void main (String [] args) {
```

```
try {
```

```
    WebServer server = new WebServer(80);
```

```
    server.addHandler("sample", new JavaServer());
```

```
    server.start();
```

```
} catch (Exception exception) {
```

```
    System.err.println("JavaServer: " + exception);
```

```
}
```

```
}
```

```
}
```

- Đối tượng lớp WebServer được khởi tạo với số hiệu cổng (ở đây là: 80).
- Server bắt đầu lắng nghe ở cổng 80.

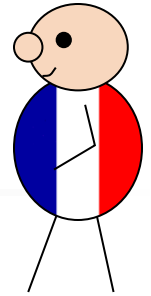




Client



Server



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodResponse>
  <params>
    <param>
      <value><int>30</int></value>
    </param>
  </params>
</methodResponse>
```