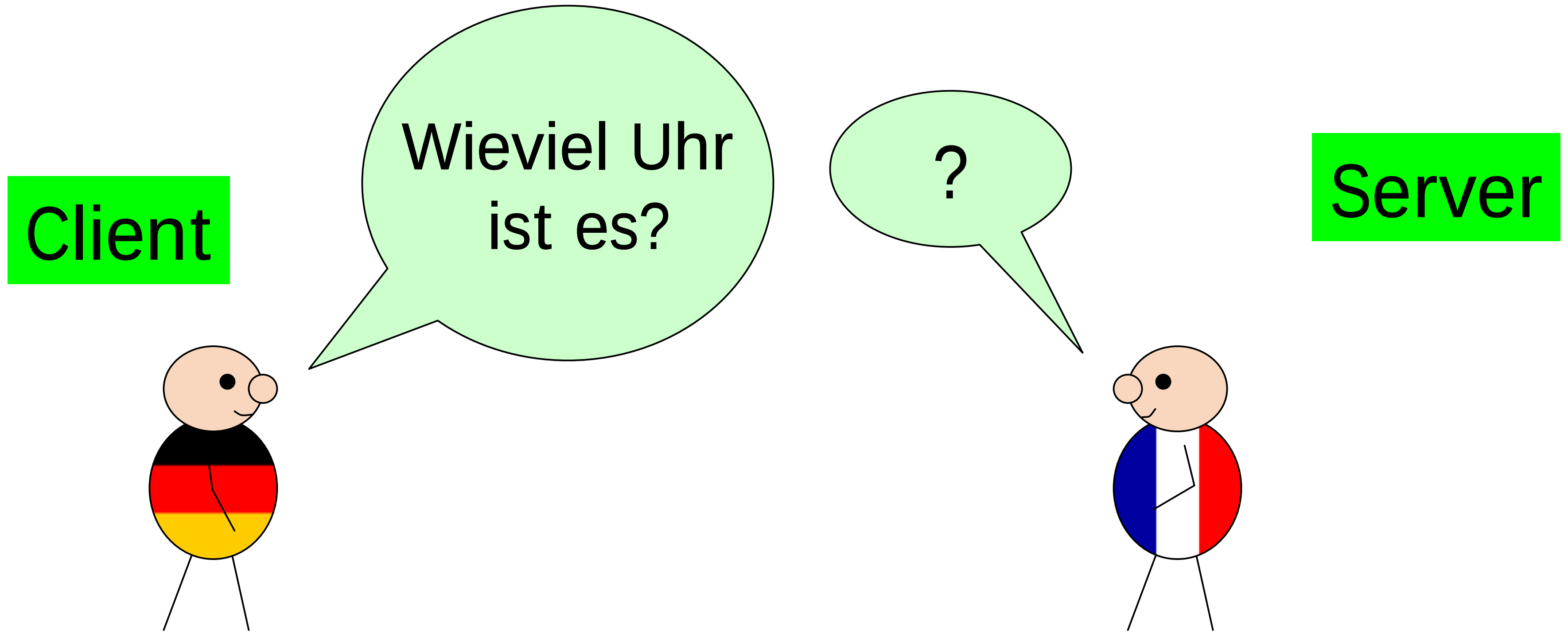
Distributed Programming - xmlrpc

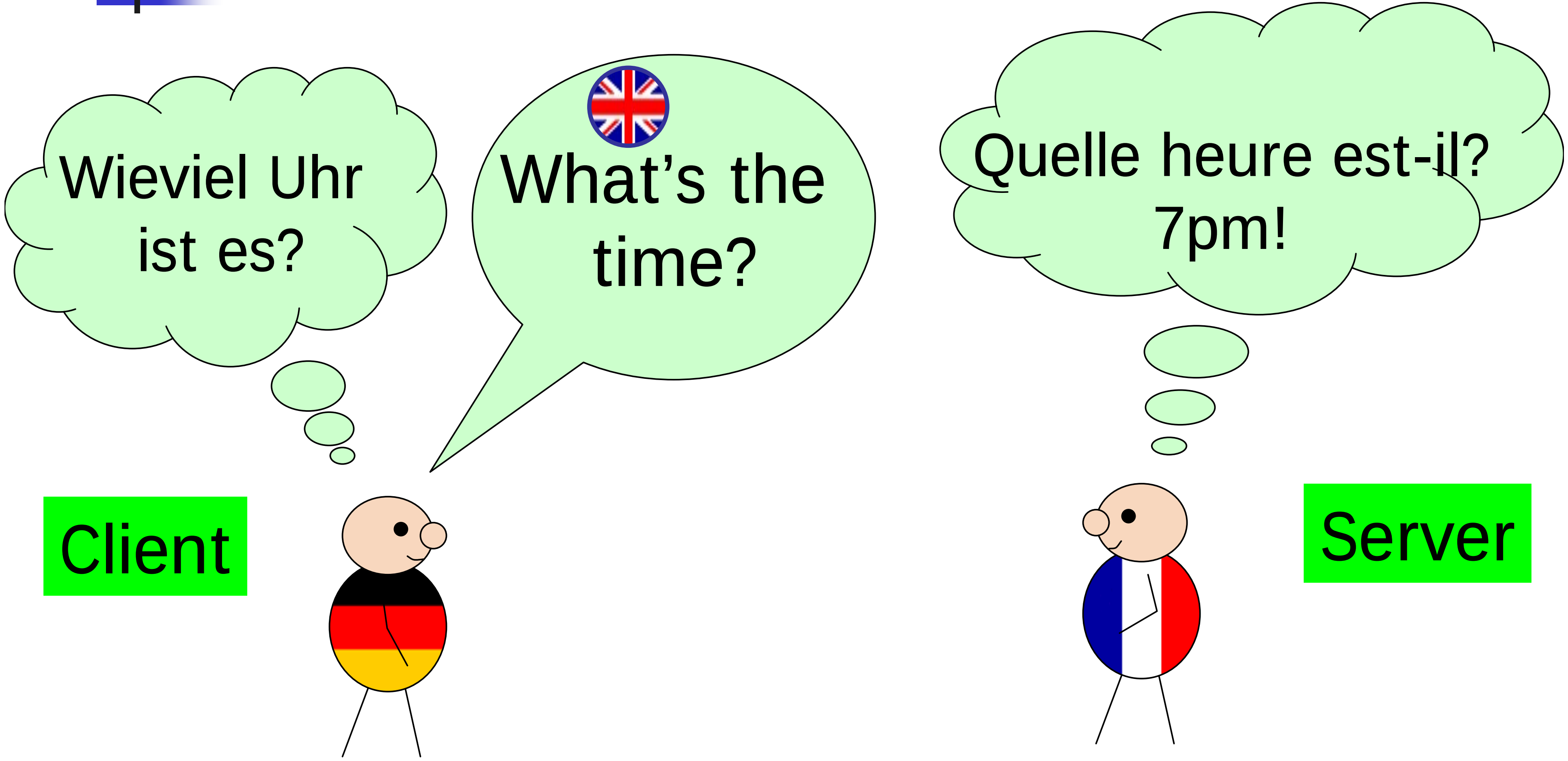
- Marc Conrad
  - D104a (Park Square Building)
  - Marc.Conrad@luton.ac.uk

  - Resources:
    - www.xmlrpc.com
    - Blackboard

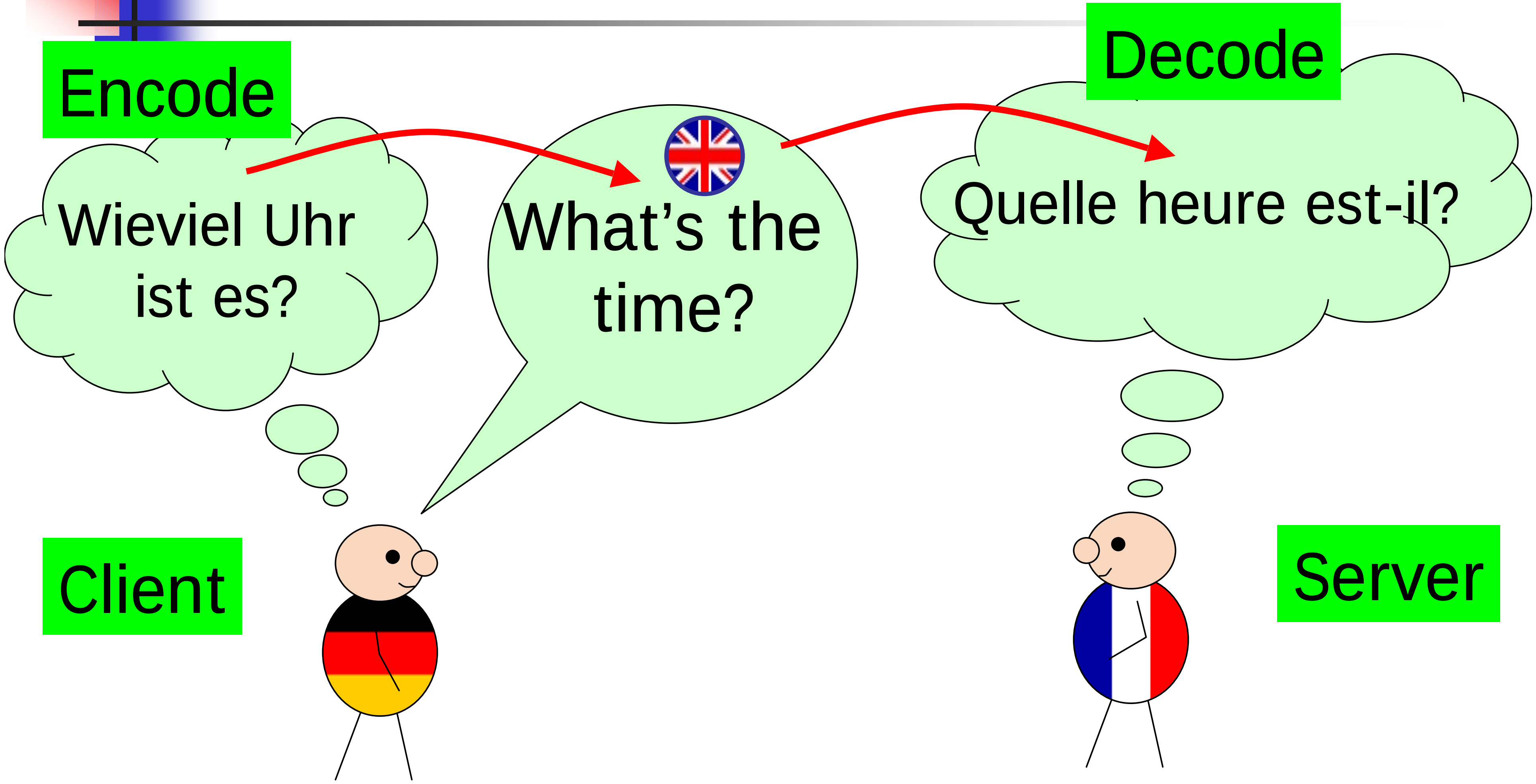
Client and Server have to understand each other.



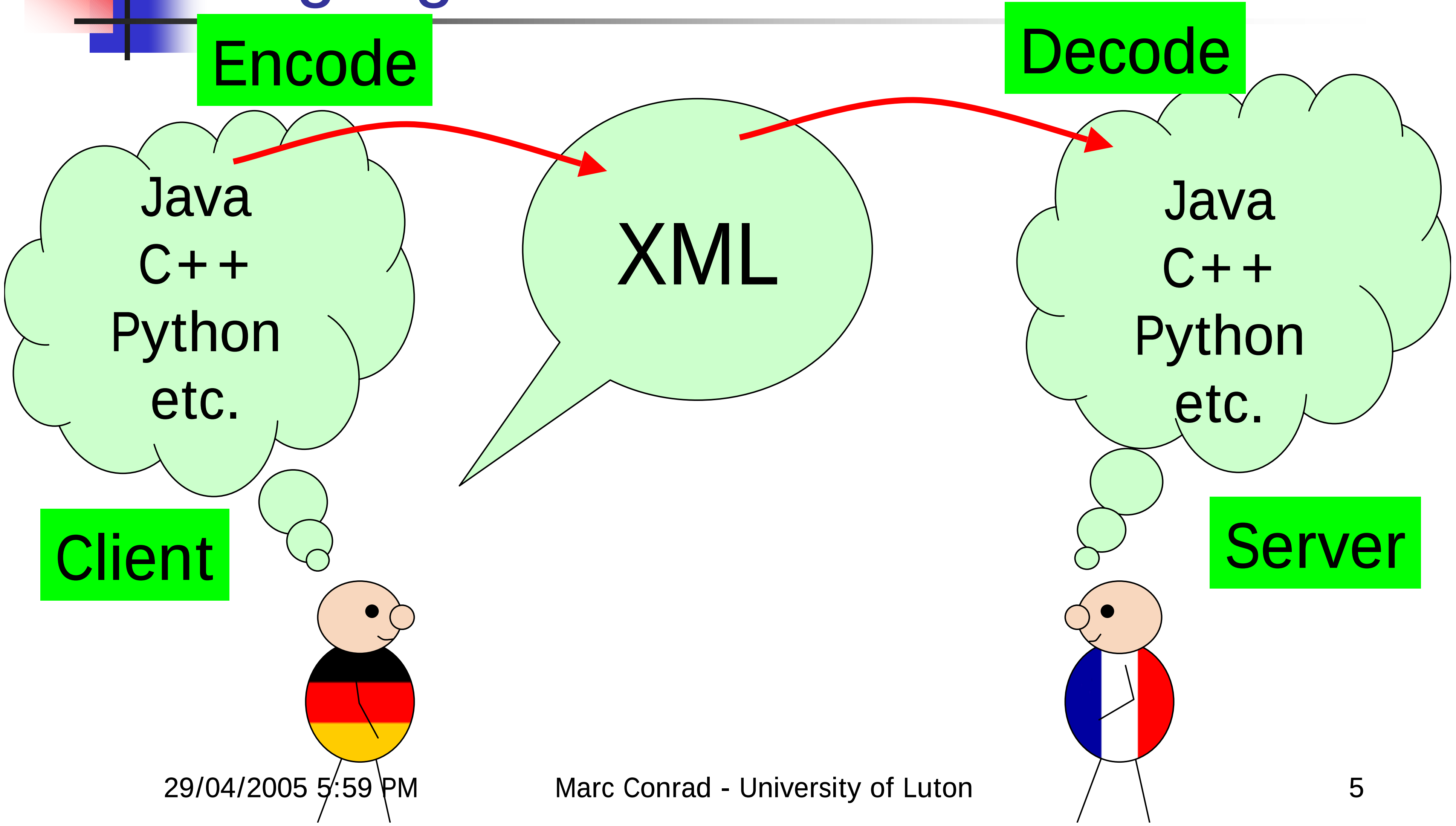
Client and Server agree on a common language.

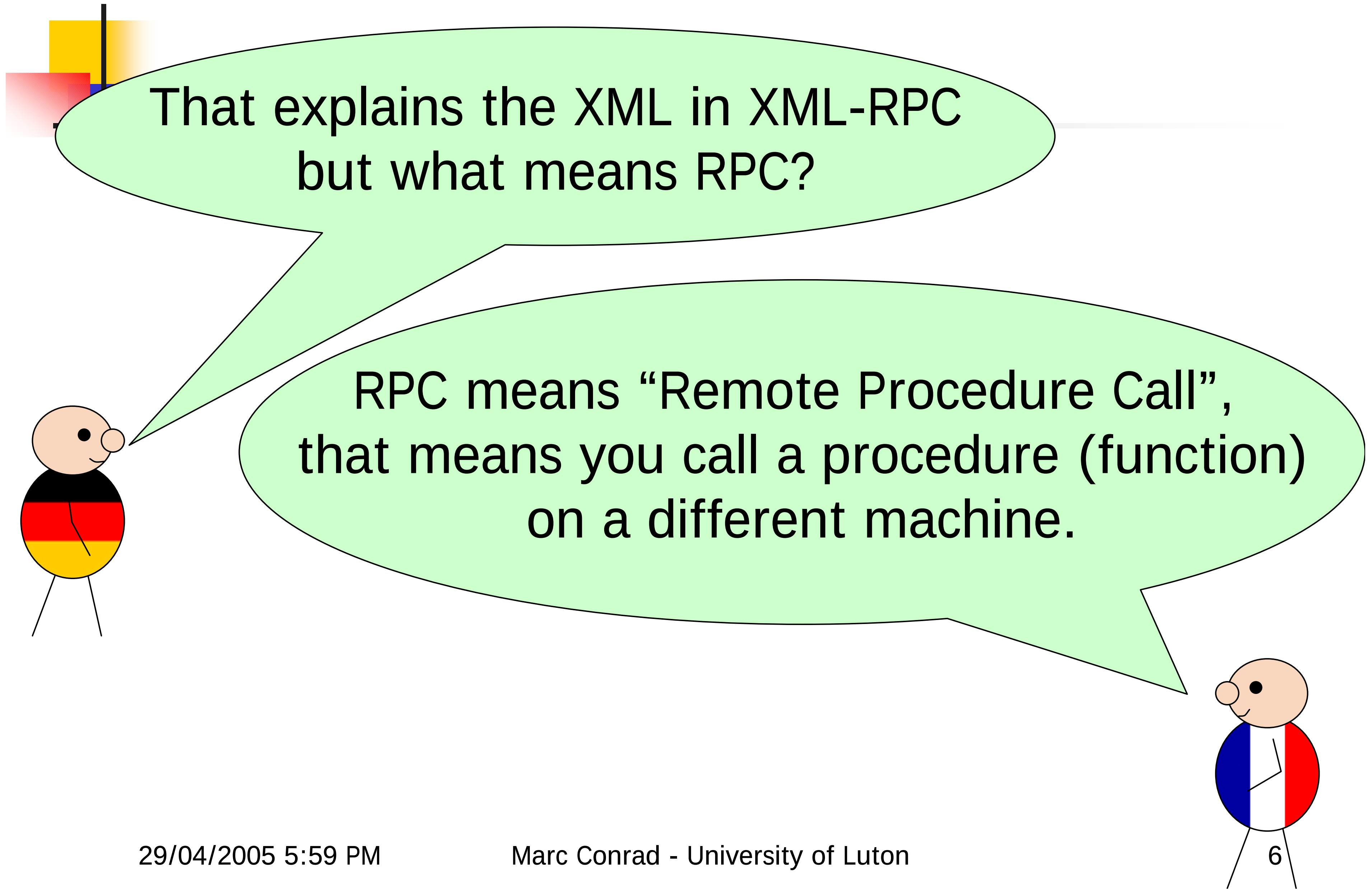


The client encodes the information in the "common language", the server decodes the information.



XML-RPC uses XML as a common language to transmit data




That explains the XML in XML-RPC but what means RPC?
RPC means “Remote Procedure Call”, that means you call a procedure (function) on a different machine.

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);
 }
}
```

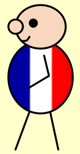
Remote procedure call (RPC)

Client

```
[…]
eg.sum(13,17);
[…]
```

Server

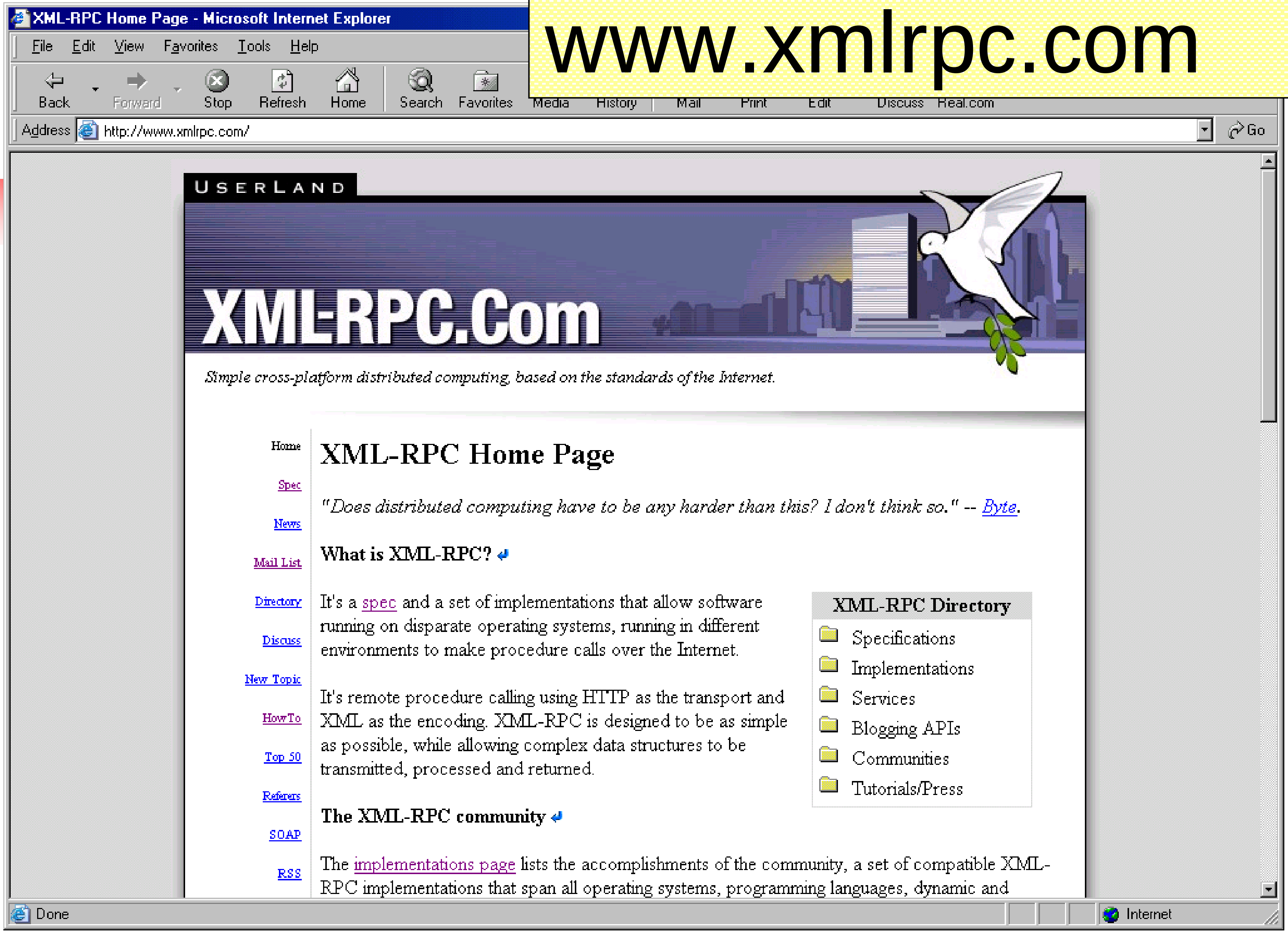
```
[…]
public int sum(int a, int b) {
    return a+b;
    }
[…]
```



RPC - Remote Procedure Call

- RPC is a powerful technique for constructing distributed, **client-server** based applications.
- It is based on extending the notion of conventional, or local procedure calling.
- As “remote” suggests, the called procedure need not to exist in the same address space as the calling procedure.
  - The two processes may be on the same system, or they may be on different systems with a network connecting them.
- By using RPC, programmers of distributed applications avoid the details of the interface with the network.

www.xmlrpc.com



XML-RPC Home Page - Microsoft Internet Explorer

File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss Real.com
Address http://www.xmlrpc.com/
Go
USERLAND
XML-RPC.Com
Home
Spec
News
Mail List
Directory
Discuss
New Topic
HowTo
Top 50
Referers
SOAP
RSS
Done
Internet

Simple cross-platform distributed computing, based on the standards of the Internet.

XML-RPC Home Page

"Does distributed computing have to be any harder than this? I don't think so." -- Byte.

What is XML-RPC?

It's a spec and a set of implementations that allow software running on disparate operating systems, running in different environments to make procedure calls over the Internet.

It's remote procedure calling using HTTP as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned.

XML-RPC Directory

- Specifications
- Implementations
- Services
- Blogging APIs
- Communities
- Tutorials/Press

The XML-RPC community

The implementations page lists the accomplishments of the community, a set of compatible XML-RPC implementations that span all operating systems, programming languages, dynamic and

What is XML-RPC?

- It's a spec and a set of implementations that allow software running on disparate operating systems, running in different environments to make procedure calls over the Internet
- It's remote procedure calling using HTTP as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned.

The specification, < 1800 words, is lightweight and easy to learn. Contains many examples.

- It's a spec and a set of implementations that allow software running on disparate operating systems, running in different environments to make procedure calls over the Internet
- It's remote procedure calling using HTTP as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned.

- It's a spec and a set of implementations that allow software running on disparate operating systems, running in different environments to

rnet

HTTP as

ling. XML-

possible,

ires to be

d.

Languages include:
C/C++, Java, Perl, Python, Frontier, Lisp, PHP, Microsoft .NET, Rebol, Real Basic, Tcl, Delphi, WebObjects and Zope

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

- It's a spec and a set of implementations that allow software running on disparate operating systems, running in different environments to make procedure calls over the Internet
- It's remote procedure calling using HTTP as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned.

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

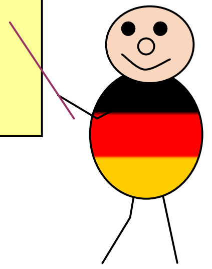
- allow software running on disparate operating systems, running in different environments to make procedure calls over the Internet
- It's remote procedure calling using HTTP as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned.

Example: An XML-RPC client/server application in Java.

- The Java package org.apache.xmlrpc provides classes to implement an XML-RPC client and an XML-RPC server. The package can be found at http://ws.apache.org/xmlrpc/
- A copy of the package is under the name cis69mc.jar on Blackboard.
- To compile and run Java classes with the package, copy it to your working directory and use the following commands (in a DOS shell):
  - javac -classpath ”cis69mc.jar;." xyz.java
  - java -classpath ”cis69mc.jar;." xyz.java

  (replace xyz by the name of your file)

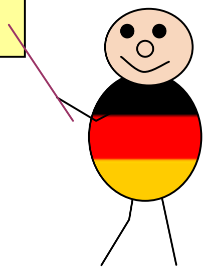
A Java Client



```
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);
   }
  }
}
```

A Java Client

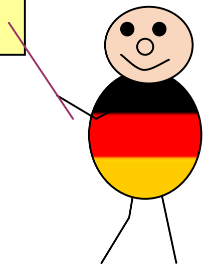


```
import java.util.*;
import org.apache.xmlrpc.*;

public class JavaClient {
 public static void main (String [] args) {
  try {
   XmlRpcClient server =
    XmlRpcClient("http:/
   Vector params = new
   params.addElement(r
   params.addElement(r
   Object result = serve
   int sum = ((Integer) r
   System.out.println("T
  } catch (Exception exc
   System.err.println("Ja
   }
  }
}
```

- The Java package org.apache.xmlrpc contains classes for XML-RPC Java clients and XML-RPC server. E.g. XmlRpcClient.
- The package java.util is necessary for the Vector class.

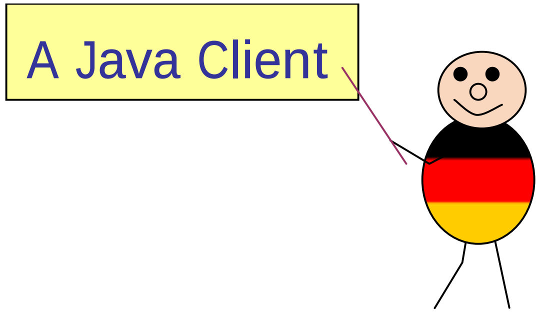
A Java Client



```
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
   params.addEler
   params.addEler
   Object result =
   int sum = ((Int
   System.out.prin
  } catch (Excepti
   System.err.prin
   }
  }
}
```

- The Java package org.apache.xmlrpc contains classes for XML-RPC Java clients and XML-RPC server. E.g. XmlRpcClient. The source code of this package is free.
- The package java.util is necessary for the Vector class. java.util.Vector is part of the Java distribution.



```
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();
```

- This line sends the request to the server. The procedure sum(17,13) is called on the server as if it were a local procedure. The return value of a procedure call is always an Object.
- “sample” denotes a *handler* that is defined in the server.

A Java Client

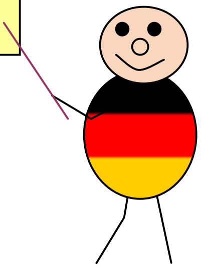
```
import java.util.*;
im

pu
 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();
```

- The parameters of the procedure call are *always* collected in a Vector.

- This line sends the request to the server. The procedure sum(17,13) is called on the server as if it were a local procedure. The return value of a procedure call is always an Object.
- “sample” denotes a *handler* that is defined in the server.

A Java Client



```
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.adc
   params.adc
   Object resu
   int sum = (
   System.out
  } catch (Exc
   System.err.
   }
  }
}
```

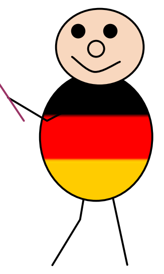
- The XmlRpcClient class is constructed by specifying the “web address” of the server machine followed by /RPC2. E.g.
  - localhost - means the local machine.
  - An IP number, e.g. 194.80.215.219
  - A name, e.g. cis69.dyndns.org
  - All of the above, followed by a port number, e.g. cis69.dyndns.org:8080. The default port is 80.

A Java Client

- As the result of the remote procedure call is always an Object it has to be casted to the appropriate type (here: 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);
  }
 }
}
```

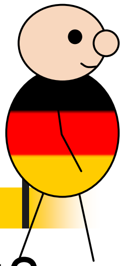
A Java Client



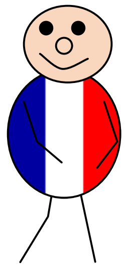
```
import java.util.*;
import org.apache.xmlrpc.*;

public class JavaClient {
 public static void main (String [] args) {
  try {
   XmlRpcClient server = new X
   Vector params = new Vector
   params.addElement(new Inte
   params.addElement(new Inte
   Object result = server.execut
   int sum = ((Integer) result).i
   System.out.println("The sum is: "+sum);
  } catch (Exception exception) {
   System.err.println("JavaClient: " + exception);
   }
  }
}
```

- When problems occur (no connection, etc.) an Exception is thrown and has to be caught.



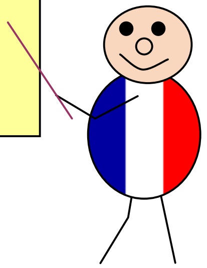
Client → Server



```
<?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>
```

- This is what the client sends to the server.

A Java Server



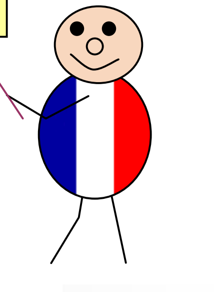
```
import org.apache.xmlrpc.*;

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

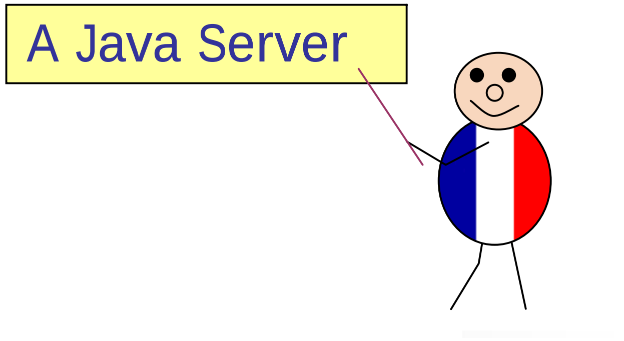
```
import org.apache.xmlrpc.*;

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());
   server.start();
  } catch (Exception exception) {
   System.e
   }
  }
 }
```

A Java Server



- The package org.apache.xmlrpc contains the class WebServer for a XML-RPC Server implementation



```
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());
   se
  } c
   Sy
   }
  }
 }
```

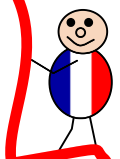
- The procedure that is called remotely is implemented as a public method in a class.
- An instance of this class is then associated with a handler that is accessible by the client.

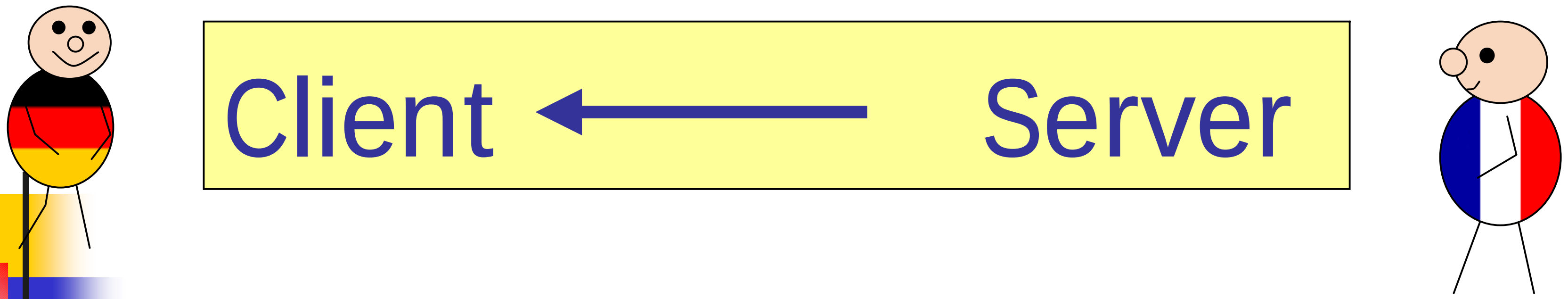
A Java Server

- The server is initialised by the port number (here: 80).
- The server starts to listen at port 80.

```
import org.ap

public class J
 public Integ
  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);
   }
  }
 }
```





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

SOAP (Simple Object Access Protocol)

- SOAP is another protocol for Client/Server applications.
- The general principle is similar as XML-RPC by using XML as common language.
- Also labelled as “lightweight”, but the specification is > 77000 words.