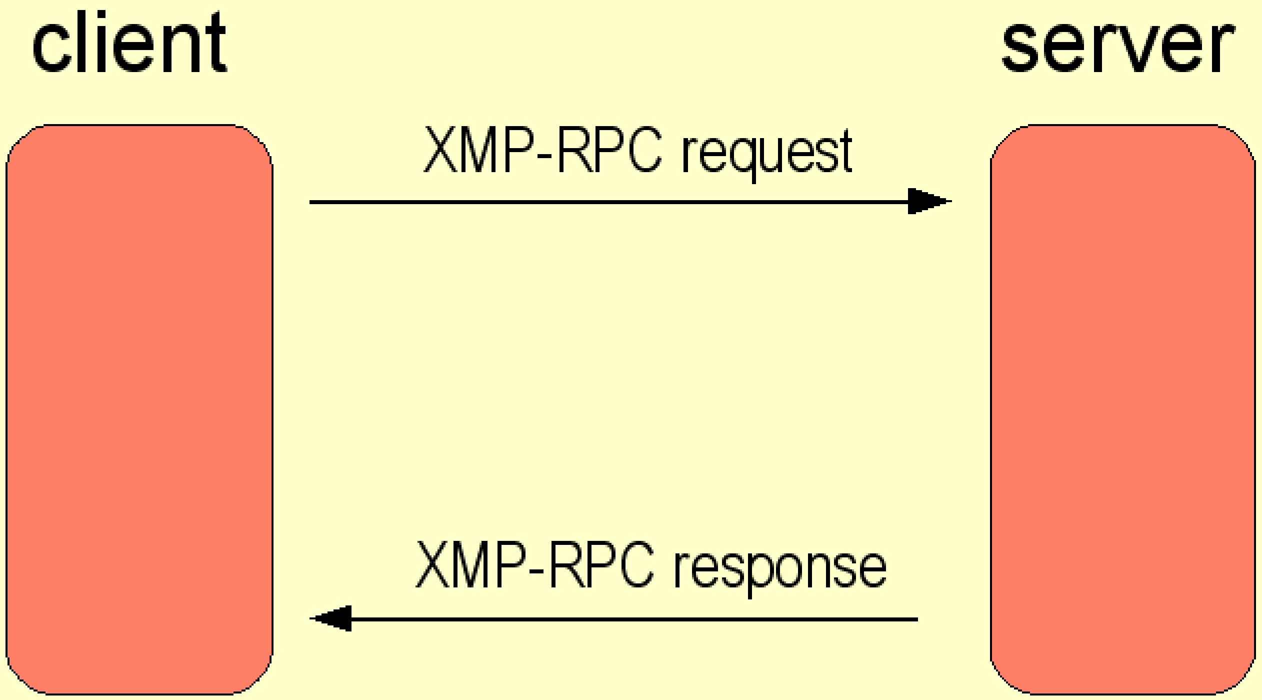
Distribuované systémy a výpočty

X36DSV

Jan Janeček

XML-RPC



jednoduchá definice

www.xmlrpc.com

implementace obvykle opřené o XML parser

XML-RPC jednoduché datové typy

boolean

```
<boolean> 1 </boolean>
```

int

```
<i4> 42 </i4>
<int> 42 </int>
```

double

```
<double> -12.53 </double>
```

string

```
<string> Hello world! </string>
```

dateTime.iso8601

```
<dateTime.iso8601>
    19980717T14:08:55
</dateTime.iso8601>
```

XML-RPC složené datové typy

array

```
<array>
    <data>
        <value><i4>1404</i4></value>
        <value>
            <string>Something here</string>
        </value>
        <value><i4>1</i4></value>
    </data>
</array>
```

struct

```
<struct>
    <member>
        <name>foo</name>
        <value><i4>1</i4></value>
    </member>
    . . .
</struct>
```

XML-RPC rozšiřující datové typy

base64

```
<base64>
        eW91IGNhbid0IHJlYWQgdGhpcyE=
</base64>
```

nil

```
<nil/>
```

XML-RPC request

```
POST /RPC2 HTTP/1.0
User-Agent: . . .
Host: . . .
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
    <param>
        <value><i4>40</i4></value>
    </param>
  </params>
</methodCall>
```

XML-RPC response

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 2005 19:55:08 GMT
Server: . . .

<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
        <value><string>South Dakota</string></value>
    </param>
  </params>
</methodResponse>
```

XML-RPC fault

```
<?xml version="1.0"?>
<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultCode</name>
          <value><int>4</int></value>
        </member>
        <member>
          <name>faultString</name>
          <value><string>Too many parameters.</string></value>
        </member>
      </struct>
    </value>
  </fault>
</methodResponse>
```

XML-RPC implementace

jednoduchá implementace

- i bez web serveru,
- jako samostatný program (standalone)
- ale často se opírá o běžný (pomalý) XML parser

implementace v řadě jazyků

Perl
Python
C / C++
Java
Ruby

možnost vzájemného kombinování

Apache XML-RPC

Client

```
public class JavaClient {
    private final static String server_url =
        "http://xmlrpc-c.sourceforge.net/api/sample.php";
    public static void main (String [] args) {
        try {
            XmlRpcClient server = new XmlRpcClient(server_url);
            Vector params = new Vector();
            params.addElement(new Integer(5));
            params.addElement(new Integer(3));
            Hashtable result = (Hashtable) server.execute
                                        ("sample.sumAndDifference", params);
            int sum = ((Integer) result.get("sum")).intValue();
            int difference = ((Integer) result.get("difference")).intValue();
```

Apache XML-RPC

Client – continuation

```
            System.out.println("Sum: " + Integer.toString(sum) +
                                ", Difference: " +
                                Integer.toString(difference));
        } catch (XmlRpcException exception) {
            System.err.println("JavaClient: XML-RPC Fault #" +
                                Integer.toString(exception.code) + ": " +
                                exception.toString());
        } catch (Exception exception) {
            System.err.println("JavaClient: " + exception.toString());
        }
    }
}
```

Apache XML-RPC

Standalone server

```
public class JavaServer {

    public JavaServer () {
        // Our handler is a regular Java object. It can have a
        // constructor and member variables in the ordinary fashion.
        // Public methods will be exposed to XML-RPC clients.
    }

    public Hashtable sumAndDifference (int x, int y) {
        Hashtable result = new Hashtable();
        result.put("sum", new Integer(x + y));
        result.put("difference", new Integer(x - y));
        return result;
    }
```

Apache XML-RPC

Standalone server – continuation

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

            // Invoke me as <http://localhost:8080/RPC2>.
            WebServer server = new WebServer(8080);
            server.addHandler("sample", new JavaServer());

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