



Orientation in Objects

## Einführung in Webservices mit XML-RPC

) Schulung )

### AUTOR

---



**Thomas Bayer**  
Orientation in Objects GmbH

) Beratung )

Veröffentlicht am: 2.12.2002

### ABSTRACT

---

XML-RPC ist ein einfaches Protokoll für entfernte Funktionsaufrufe über das Internet. Die XML-RPC Spezifikation, die neben Beispielen auch eine FAQ enthält, paßt auf 7 DIN-A 4 Seiten und wurde von Dave Winer von Userland verfaßt. XML-RPC ist leicht zu verstehen und anzuwenden. Der Footprint einer XML-RPC Implementierung kann sich auf wenige Kilobyte beschränken. Dies macht XML-RPC auch für Mobile Web Services interessant.

) Entwicklung )

) Artikel )

**Orientation in Objects GmbH**

Weinheimer Str. 68  
D-68309 Mannheim

Tel. +49 (0) 6 21 - 7 18 39 - 0  
Fax +49 (0) 6 21 - 7 18 39 - 50

www.oio.de info@oio.de

Java, XML, UML, XSLT, Open Source, JBoss, SOAP, CVS, Spring, JSF, Eclipse

## EINLEITUNG

XML-RPC ermöglicht entfernte Funktionsaufrufe über das HTTP Protokoll. Für entfernte Funktionsaufrufe gibt es die Bezeichnung *Remote Procedure Call* oder kurz [RPC](#). XML dient zur Codierung der Übergabeparameter an die entfernten Procedures. Daher der Name XML-RPC. XML-RPC ist sprachunabhängig, Server und Clients können in beliebigen Sprachen wie Java, Perl oder Ruby implementiert werden.

Ein Aufruf wird per HTTP-POST an den Server geschickt. Dieser wertet das im *Body* enthaltene XML Dokument aus und verwendet dessen Inhalt als Parameter für den Aufruf der gewünschten Funktion. Das Ergebnis wird wieder in XML gepackt und zum Client übertragen. Neben primitiven Datentypen wie *String*, *Double* und *DateTime* werden auch komplexe Typen unterstützt, die aus den primitiven Typen zusammengesetzt werden können.

## BEISPIEL

Das Beispiel eines Warenkorbes verdeutlicht die Funktionsweise und Entwicklung mit XML-RPC. Ein Server stellt eine Schnittstelle über XML-RPC für den Zugriff auf ein Warenkorb-Objekt bereit. Zunächst betrachten wir die HTTP Anfragen und Antworten, die ausgetauscht werden. Weiter unten wird der für das Beispiel notwendige Code für Client und Server erläutert.

Listing 1 zeigt den *Request* eines Clients. Wie aus der ersten Zeile ersichtlich, wird der *Request* mit der *HTTP Methode POST* an den Server übertragen. Als Empfänger wird der *Context /RPC2* verwendet. Der Body des *Requests* enthält ein XML Dokument mit dem Root-Element "methodCall". Die aufzurufende Methode wird über das Element "methodName" angegeben. Die Schreibweise läßt vermuten, dass die Methode "addPosition" auf ein Objekt "warenkorb" aufgerufen wird. Es handelt sich bei der Angabe von "warenkorb" allerdings nicht um eine Objektreferenz.

```
POST /RPC2 HTTP/1.0
User-Agent: Apache XML-RPC 1.0
Host: dchp174.oio.de
Content-Length: 221
Content-Type: text/xml
<?xml version="1.0"?>
<methodCall>
  <methodName>warenkorb.addPosition</methodName>
  <params>
    <param>
      <value>Dauerlutscher</value>
    </param>
    <param>
      <value>
        <int>10</int>
      </value>
    </param>
    <param>
      <value>
        <double>0.38</double>
      </value>
    </param>
  </params>
</methodCall>
```

### Beispiel 1: XML-RPC Request

Mit XML-RPC können nur Procedures und keine Methoden aufgerufen werden, da für den Aufruf einer Methode auch ein Objekt, auf dem die Methode ausgeführt werden soll, übergeben werden muß. Der Anbieter eines XML-RPC Web Services kann die entfernte Funktion selbstverständlich auch über eine statische Methode realisieren.

Das nächste Element "params" enthält die Parameter des Aufrufs. Parameter bekommen in XML-RPC keinen Namen. Allein die Position eines Parameters entscheidet über dessen Bedeutung.

Beim Aufruf von "addPosition" wird die Artikelbezeichnung, die Menge und der Preis übergeben.

Der Server nimmt die Anfrage entgegen und leitet den Aufruf an die entsprechende Funktion bzw. Methode weiter. Die Antwort enthält, wie Listing 2 zeigt, einen Wert, der die Anzahl der zum Warenkorb hinzugefügten Positionen angibt.

```
HTTP/1.0 200 OK
Server: Apache XML-RPC 1.0
Connection: close
Content-Type: text/xml
Content-Length: 135
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <int>1</int>
      </value>
    </param>
  </params>
</methodResponse>
```

### Beispiel 2: Die Antwort des Servers

Über die Reihenfolge und Bedeutung der Parameter muss zwischen Client und Server Einverständnis bestehen. Die Schnittstelle des Servers kann nicht evolutionär verändert und erweitert werden.

## KOMPLEXE TYPEN

XML-RPC unterstützt die Übergabe von Parametern mit zusammengesetzten Typen. Listing 3 zeigt eine Anfrage nach den Positionen des Warenkorbes. Der Warenkorb kann mehrere Positionen enthalten, die jeweils Artikel, Menge und Preis beinhalten.

```
POST /RPC2 HTTP/1.0
User-Agent: Apache XML-RPC 1.0
Host: dchp174.oio.de
Content-Length: 67
Content-Type: text/xml
<methodCall>
  <methodName>warenkorb.getPositionen</methodName>
</methodCall>
```

### Beispiel 3: Frage nach dem Inhalt des Warenkorbes

Abbildung 1 zeigt, wie die Positionen des Warenkorbes mit den in XML-RPC zur Verfügung stehenden Datentypen abgebildet werden können. Ein übergeordnetes Array nimmt jeweils ein Array für eine Position und deren Werte auf.

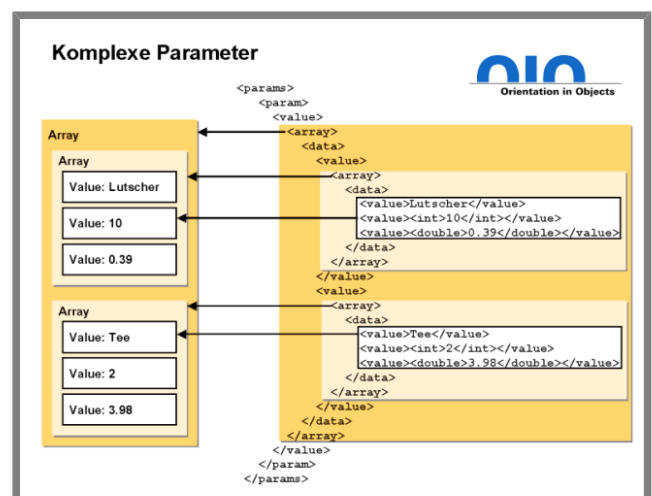


Abbildung 1: Positionen des Warenkorbes als Parameter

Die Antwort des Servers mit den codierten Positionen zeigt Listing 4.

```

HTTP/1.0 200 OK
Server: Apache XML-RPC 1.0
Connection: close
Content-Type: text/xml
Content-Length: 405
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <array>
          <data>
            <value>
              <array>
                <data>
                  <value>Lutscher</value>
                  <value>
                    <int>10</int>
                  </value>
                  <value>
                    <double>0.39</double>
                  </value>
                </data>
              </array>
            </value>
          </param>
          <param>
            <value>
              <array>
                <data>
                  <value>Tee</value>
                  <value>
                    <int>2</int>
                  </value>
                  <value>
                    <double>3.98</double>
                  </value>
                </data>
              </array>
            </value>
          </param>
        </params>
      </methodResponse>

```

Beispiel 4: Ausgabe des Warenkorbes

## UNTERSTÜTZTE DATENTYPEN

XML-RPC unterstützt die in Tabelle 1. aufgeführten Datentypen.

Datentyp	Beschreibung	Beispiel
boolean	Ein Bit.	true, false
int	32 Bit Integer mit Vorzeichen	10, -10, 0
double	Fliesskommazahl mit doppelter Präzision	-94.4, -0.5, 0.74
string	Zeichenkette	Hossa, Bar, Foo
dateTime.iso8601	Zeit und Datum	20021105T14:14:55
array	Eindimensionales Array. Der Typ der Werte ist beliebig	
struct	Schlüssel-Wert Paare. Der Schlüssel ist ein string, der Wert kann von beliebigem Typ sein. Structs in XML-PRC entsprechen Maps oder Hashtables in Java.	
base64	Base64 encodierte Binärdaten	

Tabelle 1: Datentypen

## FEHLERBEHANDLUNG

Fehlermeldungen, die bei der Ausführung eines *Requests* auf dem Server auftreten, können dem Client wie Listing 5 zeigt mit einem *fault*-Element übergeben werden. Das *fault*-Element enthält ein *Struct* mit Fehlercode und Fehlerbeschreibung. Im Beispiel wird eine Java *Exception* in das *fault*-Element eingepackt.

Der *HTTP Status Code* ist auch im Fehlerfall "200 OK". Nur im Body der Nachricht ist der Fehler erkennbar. Für das Fehlerhandling wurde nicht auf die Mechanismen von HTTP zurückgegriffen, sondern mit dem *fault*-Element etwas eigenes geschaffen.

```

<?xml version="1.0"?>
HTTP/1.0 200 OK
Server: Apache XML-RPC 1.0
Connection: close
Content-Type: text/xml
Content-Length: 317
<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultString</name>
          <value>
            java.lang.StringIndexOutOfBoundsException:
              String index out of range: -1
          </value>
        </member>
        <member>
          <name>faultCode</name>
          <value><int>0</int></value>
        </member>
      </struct>
    </value>
  </fault>
</methodResponse>

```

Beispiel 5: Fehlermeldung

## BEISPIEL XML-RPC MIT JAVA

Das folgende Codebeispiel wurde für die Implementierung Apache XML-RPC geschrieben. Für XML-RPC gibt es kein standardisiertes API wie z.B. JAX-RPC für SOAP. Jede Implementierung verwendet ihre eigenen Schnittstellen und Klassen.

Betrachten wir zunächst den Code für den Server. Der Server besteht aus einer *main*-Methode und einer normalen Java Klasse, die den Warenkorb implementiert.

Im Code in Listing 6 wird ein kleiner Webserver gestartet, der von Apache XML-RPC mitgeliefert wird. An den Namen "warenkorb" wird anschließend ein Objekt vom Typ Warenkorb gebunden.

```

public class Server {
  public static void main(String args[])
    throws IOException {
    Warenkorb warenkorb = new Warenkorb();
    WebServer webserver = new WebServer( 5555);
    Webserver.addHandler( "warenkorb", warenkorb);
  }
}

```

Beispiel 6: XML-PRC Server

Bei der Implementierung des Warenkorbes handelt es sich um eine normale Java Klasse, die in Listing 7 abgedruckt ist. Das einzige XML-RPC spezifische an dieser Klasse ist die Methode "getPositionen", die den Rückgabewert in Vektoren einpackt, so dass XML-RPC diese in die Rückgabestruktur packen kann.

```

public class Warenkorb {
    private Vector positionen = new Vector();
    public int addPosition( String ware,
                          int menge,
                          double preis) {
        positionen.add(
            new Position( ware, menge, preis)
        );
        return 1;
    }
    public Collection getPositionen() {
        Vector ret = new Vector();
        Iterator positionIter = positionen.iterator();
        while( positionIter.hasNext() ) {
            ret.add(
                ((Position)positionIter.next()).getParams()
            );
        }
        return ret;
    }
}

```

#### Beispiel 7: Implementierung des Warenkorbes

Jetzt fehlt noch der Client, der die Methoden der Klasse Warenkorb über das Netz aufruft. Im Listing 8 wird ein *Vector* mit den Übergabeparametern gefüllt, die beim Aufruf übergeben werden. Der Aufruf erfolgt über eine Klasse "XmlRpcClient", die mit der Adresse des Servers initialisiert wurde. Beim Aufruf des Servers mit der Methode execute wird der Funktionsname auf dem Server und die Parameter übergeben.

```

XmlRpcClient xmlrpc = new
XmlRpcClient("http://hermes.oio.de:5555/RPC2");
Vector params = new Vector();
params.addElement("Lutscher");
params.addElement(new Integer(10));
params.addElement(new Double(0.39));
xmlrpc.execute("warekorb.addPosition", params);

```

#### Beispiel 8: XML-RPC Client

## PERFORMANCE

Es liegt die Vermutung nahe, dass die Verarbeitung der in XML codierten Nachrichten viel Zeit in Anspruch nimmt und daher die Performance stark beeinträchtigt wird. Die Struktur der Nachrichten ist im Gegensatz zu SOAP Messages einfach. Es fehlen u.a. Angaben für Namespaces und für das Encoding. Das Parsen der XML Nachrichten kann über kompakte und schnelle Parser geschehen, die weder Namespaces, noch Attribute oder DTDs kennen. In der Praxis ist daher XML-RPC recht performant. Die *Roundtrip* -Zeiten für entfernte Aufrufe liegen meist bei wenigen Millisekunden. Der Speicher und CPU Verbrauch für XML-RPC ist aufgrund der Größe zu vernachlässigen.

## IMPLEMENTIERUNGEN

Zahlreiche XML-RPC Implementierungen sind für Unix, Linux, Windows und Mac verfügbar. Die Liste zählt nur wenige der verfügbaren Implementierungen auf. Allein für PHP sind mehr als fünf Implementierungen verfügbar.

- [.NET \(XML-RPC.NET von Cook Computing\)](#)
- [Apache XML-RPC](#)
- [Perl \(XML::RPC\)](#)
- [Phyton \(xmlrpclib von Secret Labs\)](#)
- [Ruby \(xmlrpc4r - XML-RPC for Ruby\)](#)
- [Tcl \(XML-RPC for Tcl\)](#)

## APACHE XML-RPC

Apache XML-RPC ist eine kompakte Implementierung für Java. Zusätzlich zum Standard gibt es Erweiterungen für Servlets und SSL. Ein Web Container wie Jakarta Tomcat kann als Server dienen oder es kann ein eingebauter spezieller HTTP Server verwendet werden.

Für die Einbindung von Apache XML-RPC genügt ein 57 KByte grosses Jar-File, welches bereits den XML Parser MinML enthält. Apache XML-RPC eignet sich aufgrund der geringen Größe auch für den mobilen Einsatz auf dem Handy mit der Java Micro Edition.

## ANWENDUNGEN MIT XML-RPC UNTERSTÜTZUNG

Eine Reihe von Anwendungen bietet Schnittstellen über XML-RPC an. Neben den Produkten von Userland wie Frontier, Manila und Radio unterstützen Zope oder KDE ebenfalls XML-RPC.

## GRENZEN VON XML-RPC

XML-RPC Implementierungen verwenden oft kleine kompakte XML Parser wie MinML. Nicht immer kann davon ausgegangen werden, dass der verwendete Parser auch Unicode unterstützt.

Für XML-RPC gibt es keine Schnittstellenbeschreibungen im Stil von WSDL oder CORBA IDL. Proxys für den Zugriff auf den Server können daher nicht automatisiert erzeugt werden.

Bei vielen Implementierungen müssen Objekte von Hand in Strukturen aus Arrays und Struts (Maps oder Hashtables) übersetzt werden. Das automatische Serialisieren hat seine Grenzen.

XML-RPC ist als endgültiger Standard anzusehen, der sich in nächster Zeit nicht mehr weiter entwickeln wird.

## FAZIT

Kommunikationspartner in anderen Organisationen, die Dienste bereitstellen oder Dienste verwenden möchten, beeinflussen neben den zur Verfügung stehenden Entwicklungstools die Entscheidung für ein Webservice Protokoll. Im Online-Publishing, bei Weblogs und Microcontent ist XML-RPC stark vertreten, während SOAP im eBusiness Umfeld dominiert.

Die Schlantheit und Unkompliziertheit von XML-RPC ist zugleich Vorteil und Nachteil. Wer schnell und unkompliziert Web Services realisieren möchte, sollte das Protokoll XML-RPC ins Auge fassen.

## RESSOURCEN

---

- XML-RPC Home Page  
[\(http://www.xmlrpc.com/\)](http://www.xmlrpc.com/)
- Apache XML-RPC  
[\(http://xml.apache.org/xmlrpc/\)](http://xml.apache.org/xmlrpc/)
- MinML a minimal XML parser  
<http://www.wilson.co.uk/xml/minml.htm>