27.5.2 Klasse RpcClient

Ein Objekt der Klasse RpcClient (gb.xml.rpc) können Sie so erzeugen:

```
Dim hRpcClient As RpcClient
hRpcClient = New RpcClient(remoteFunction As RpcFunction)  As "EVENT-NAME"
```

Beachten Sie, dass Sie alle Eigenschaften einer (neuen) RPC-Funktion festgelegt haben müssen, bevor Sie ein neues RPC-Client-Objekt erzeugen.

27.5.2.1 Eigenschaften

Die Klasse *RpcClient* verfügt über drei Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Mode	Integer	Gibt den Modus für den intern verwendeten HTTP-Server an oder setzt diese Eigenschaft. Als Werte für die Eigenschaft Modus können diese drei Konstanten verwendet werden: Const offLine As Integer = 0, Const httpSync As Integer = 1 und Const httpAsync As Integer = 2.
RpcMethod	RpcFunction	Diese Eigenschaft kann nur gelesen werden und gibt eine RpcFunktion zurück.
URL	String	Der URL besteht aus dem Locator des Servers mit nachfolgender Port-Nummer, auf dem der XML-RPC-Server lauscht. Beispiel → localhost:1088

Tabelle 27.5.2.1.1 : Eigenschaften der Klasse RpcClient

27.5.2.2 Methoden

Die Klasse *RpcClient* verfügt nur über diese zwei Methoden:

Methode	Rückgabetyp	Beschreibung
Function Call(Data As Variant[])	Variant	Zurückgegeben wird der Funktionswert des Methodenaufrufs. Data ist ein Variant-Array mit den Argumenten, die an den Server übergeben werden.
Function EvalReply(sCad As String)	Variant	Gibt den Datenteil einer XML-RPC-Antwort zurück. Diese Daten können u.a. vom Typ Integer, Float, String, Boolean, Array oder Struct sein.

Tabelle 27.5.2.2.1 : Methoden der Klasse RpcClient

27.5.2.3 Client-Projekt 1

Als Server für das erste Projekt wird ein XML-RPC-Server → Kapitel 27.5.1 RPC-Server gestartet, der seine Dienste auf einem PC lokal auf dem Port 1088 anbietet. Der Port 1088 wurde gewählt, weil er einerseits über den ersten Standard-Ports (>1023) liegt und andererseits offiziell nicht benutzt wird oder reserviert ist. Über die RPC-Schnittstelle können die folgenden *vier* Funktionen vom Client aufgerufen werden:

- *system.methodHelp* – Die Methode 'system.methodHelp' gibt den Hilfetext zurück, sofern ein Hilfe-Text für die spezielle Methode mit *RpcServer.Help* definiert ist. Ansonsten gibt sie einen leeren String zurück.

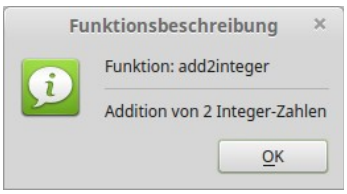

Abbildung 27.5.2.3.1: Dokumentation der Funktion add2integer

- *system.methodSignature* – Die Methode 'system.methodSignature' gibt ein Array von bekannten Signaturen (als ein Array von Arrays) für den speziellen Methodennamen zurück. Wenn keine Signaturen bekannt sind, gibt sie ein leeres Array zurück.

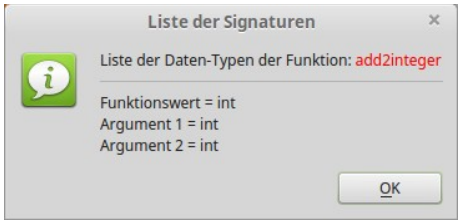

Abbildung 27.5.2.3.2: Liste der Signatur der Funktion add2integer

- *system.listMethods* – Die Methode 'system.listMethods' listet alle Methoden als Elemente in einem Array auf, die für die RPC-Schnittstelle auf dem XML-RPC-Server implementiert sind. Der Aufruf erfolgt ohne Argumente - das Array der Argumente ist leer.

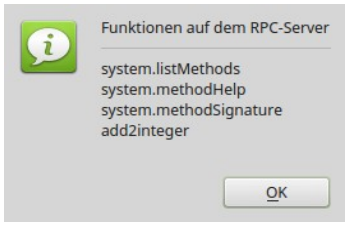

Abbildung 27.5.2.3.3: Liste der implementierten Funktionen

- *add2integer* – Diese Methode liefert als Funktionswert die Summe von zwei als Argument übergebene Integer-Zahlen zurück.

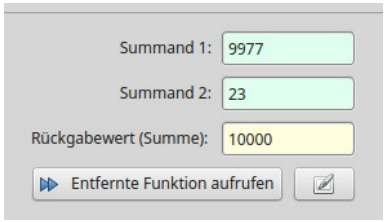

Abbildung 27.5.2.3.4: Aufruf der Methode add2integer

Beachten Sie: In beiden Projekten erfolgt für die beiden Summanden keine Prüfung des Datentyps und des Integer-Bereiches.

Der Quelltext für den XML-RPC-Client wird vollständig angegeben:

```
' Gambas class file

Private $sRPCURL As String
Private hXMLRPCFunction As RpcFunction
Private hXMLRPCClient As RpcClient
Private $aArguments As Variant[]
Private $sRPCFunctionName As String

Public Sub Form_Open()
  FMain.Resizable = False
  FMain.Title = "HTTP-SERVICE * RPC-CLIENT"
  txbSumme.ReadOnly = True
' $sRPCURL = "0.0.0.0:1088"
' $sRPCURL = "192.168.2.101:1088"
' $sRPCURL = "127.0.0.1:1088"
  $sRPCURL = "localhost:1088" ' This Port officially not used or reserved
  $sRPCFunctionName = "add2integer"
End

Public Sub btnGetResult_Click()
  GetResult()
End
```

```
Public Sub btnGetInformation_Click()
  GetDoc()
  GetSignatures()
  GetMethods()
End
Public Sub txbSummand1_Change()
  txbSumme.Clear()
End

Public Sub txbSummand2_Change()
  txbSumme.Clear()
End

Private Sub GetResult()

  Dim iResult As Integer

  $aArguments = New Variant[]
  hXMLRPCFunction = New RpcFunction($sRPCFunctionName, [XmlRpc.xInteger, XmlRpc.xInteger], XmlRpc.xInteger)
  hXMLRPCClient = New RpcClient(hXMLRPCFunction) As "hXMLRPCClient"
  hXMLRPCClient.URL = "http://" & $sRPCURL

  $aArguments.Add(CInt(txbSummand1.Text))
  $aArguments.Add(CInt(txbSummand2.Text))

  Try iResult = hXMLRPCClient.Call($aArguments)
  If Not Error Then
     txbSumme.Text = Str(iResult)
  Else
     Message.Error(("Error-Text: ") & Error.Text & gb.NewLine &
                   ("Error-Location:  ") & Error.Where & gb.NewLine &
                   ("URL: '") & $sRPCURL & " '")
  Endif

End

Private Sub GetDoc()

  Dim sResult As String

  $aArguments = New Variant[]
  hXMLRPCFunction = New RpcFunction("system.methodHelp", [XmlRpc.xString], XmlRpc.xString)
  hXMLRPCClient = New RpcClient(hXMLRPCFunction) As "hXMLRPCClient"
  hXMLRPCClient.URL = "http://" & $sRPCURL

  $aArguments.Add("add2integer")

  Try sResult = hXMLRPCClient.Call($aArguments)
  If Not Error Then
     Message.Title = ("Description of the function")
     Message.Info(("Function: ") & $sRPCFunctionName & "<hr>" & sResult)
  Else
     Message.Error(("Error-Text: ") & Error.Text)
  Endif

End

Private Sub GetSignatures()

  Dim aResult As New RpcArray
  Dim sMessage As String
  Dim iIndex As Integer

  $aArguments = New Variant[]
  hXMLRPCFunction = New RpcFunction("system.methodSignature", [XmlRpc.xString], XmlRpc.xArray)
  hXMLRPCClient = New RpcClient(hXMLRPCFunction) As "hXMLRPCClient"
  hXMLRPCClient.URL = "http://" & $sRPCURL

  $aArguments.Add("add2integer")

  Try aResult = hXMLRPCClient.Call($aArguments)
  If Not Error Then
     Message.Title = ("List of data types")
     sMessage = ("List of data types of the function")
     sMessage &= "<font color='red'>" & $sRPCFunctionName & "</font>"
     sMessage &= "<hr>"
     For iIndex = 1 To aResult.Count
       If iIndex = 1 Then
          sMessage &= ("Function value ") & " = " & aResult[iIndex - 1] & gb.NewLine
```

```
      Else
         If iIndex = aResult.Count Then
            sMessage &= "Argument " & Str(iIndex - 1) & " = " & aResult[iIndex - 1]
         Else
            sMessage &= "Argument " & Str(iIndex - 1) & " = " & aResult[iIndex - 1] & gb.NewLine
         Endif
      Endif
    Next
    Message.Info(sMessage)
  Else
    Message.Error(("Error-Text: ") & Error.Text)
  Endif
End

Private Sub GetMethods()

  Dim aResult As New RpcArray
  Dim sMessage As String
  Dim iIndex As Integer

  $aArguments = New Variant[]
  hXMLRPCFunction = New RpcFunction("system.listMethods", [], XmlRpc.xArray)
  hXMLRPCClient = New RpcClient(hXMLRPCFunction) As "hXMLRPCClient"
  hXMLRPCClient.URL = "http://" & $sRPCURL

  Try aResult = hXMLRPCClient.Call($aArguments)
  If Not Error Then
    Message.Title = ("List of funktions")
    sMessage = ("Functions on the RPC server") & "<hr>"
    For iIndex = 1 To aResult.Count
      sMessage &= aResult[iIndex - 1] & gb.NewLine
    Next
    Message.Info(sMessage)
  Else
    Message.Error(("Error-Text: ") & Error.Text)
  Endif

End
```

27.5.2.4 Client-Projekt 2

Im Gegensatz zum ersten Projekt wird ein XML-RPC-Server nur dann gestartet, wenn die PHP-Datei *xmlrpc.server.php* im Web-Ordner *~/public_html* aufgerufen wird → Kapitel 27.5.1 und 24.13. Der Quelltext für den Client2 unterscheidet sich vom Client1 aus dem Projekt 1 deshalb vor allem in der unterschiedlichen Diktion des URL. Als Funktion ist ebenso wie im ersten Projekt *add2integer* implementiert.

```
' Gambas class file

Private $sRPCURL As String

Public Sub Form_Open()
  FMain.Resizable = False
  FMain.Title = "WEBSERVICE * RPC-CLIENT"
  txbSumme.ReadOnly = True
  $sRPCURL = "localhost/~" & System.User.Name &/ "xmlrpc.server.php"
End

Public Sub btnGetResult_Click()
  GetResult()
End

Public Sub txbSummand1_Change()
  txbSumme.Clear()
End

Public Sub txbSummand2_Change()
  txbSumme.Clear()
End

Private Sub GetResult()

  Dim hXMLRPCClient As RpcClient
  Dim hXMLRPCFunction As RpcFunction
  Dim aArguments As New Variant[]
  Dim iResult As Integer

  hXMLRPCFunction = New RpcFunction("add2integer", [XmlRpc.xInteger, XmlRpc.xInteger], XmlRpc.xInteger)

  hXMLRPCClient = New RpcClient(hXMLRPCFunction) As "hXMLRPCClient"
```

```
  hXMLRPCClient.URL = "http://" & $sRPCURL

  aArguments.Add(CInt(txbSummand1.Text))
  aArguments.Add(CInt(txbSummand2.Text))

  Try iResult = hXMLRPCClient.Call(aArguments)
  If Not Error Then
     txbSumme.Text = Str(iResult)
  Else
     Message.Error("Fehler-Text:    " & Error.Text & gb.NewLine &
                   "Fehler-Ort:     " & Error.Where & gb.NewLine &
                   "URL:    '" & $sRPCURL & " '")
  Endif

End
```

Das Ergebnis der algebraischen Addition wird Sie wohl nicht überraschen:

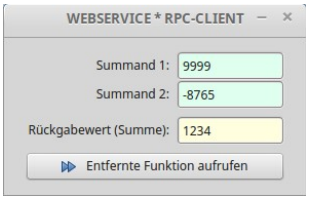

Abbildung 27.5.2.4.1: Aufruf der auf einem RPC-Server implementierten Funktion *add2integer*