XML-RPC HOWTO

KiddEric

Source Builders

eric.kidd (at) pobox.com

And Other Contributors

도운이

원영식

monpetit (at) orgio.net

0.8.0, 2001-04-12

이 문서는 다양한 컴퓨터언어로 XML-RPC 클라이언트와 서버를 만드는 방법을 기술한다. Perl, Python, C, C++, Java, PHP 등의 각종 언어로 된 예제 코드를 제공하며, XML-RPC를 지원하는 모든 오퍼레이팅 시스템의 Zope와 KDE 2.0 섹션도 포함한다.

고친 과정

고침 0.8.02001-04-12

공통 인터페이스 장 수정. XML-RPC.Net 정보 추가.

고침 0.7.02001-04-03

C++ proxy classes 장 추가.

고침 0.6.02001-02-02

Ruby, K, 공통 인터페이스 장 추가.

고침 0.5.02001-01-23

최초 버전

1. 법률적 공지(Legal Notice)

역자주: 라이센스는 원문을 그대로 싣습니다.

2. XML-RPC란 무엇인가?

XML-RPC (http://www.xmlrpc.com/)는 HTTP를 통한 간단하고 이식성 높은 원격 프로시져 호출 방법이다. XML-RPC는 Perl, Java, Python, C, C++, PHP 와 그 외 다른 많은 언어로 사용할 수 있으며 Unix, Windows, 그리고 Macintosh에서 실행할 수 있다.

다음은 Perl로 만든 간단한 XML-RPC 클라이언트 프로그램이다. (Ken MacLeod의 Frontier::Client (http://bitsko.slc.ut.us/~ken/xml-rpc/) 모듈 사용)

```
use Frontier::Client;
$server = Frontier::Client->new(url => 'http://betty.userland.com/RPC2');
$name = $server->call('examples.getStateName', 41);
print "$name\n";
```

이 프로그램을 실행하면, 원격 서버에 연결하여 미국의 주명(state name)을 얻어 출력한다. (이 예제에서 41번 주는 South Dakota 일 것이다.)

다음은 동일한 Python 프로그램이다. (Fredrik Lundh의 xmlrpclib (http://www.pythonware.com/products/xmlrpc/) 사용)

```
python> import xmlrpclib
python> server = xmlrpclib.Server("http://betty.userland.com/RPC2")
python> server.examples.getStateName(41)
'South Dakota'
```

이제부터 다양한 프로그래밍 언어로 어떻게 XML-RPC 클라이언트와 서버를 만드는지에 대하여 살펴볼 것이다.

2.1. 어떻게 동작하는가

XML-RPC는 Dave Winer의 공식 명세서 (http://www.xmlrpc.com/spec)에 상세히 정의되어 있다. 궁금한 사항이 있다면 직접 가서 보라. 간단하고 읽기 쉽게 되어 있다.

(서버와 클라이언트 간의) 통신 상에서, XML-RPC 값은 XML 코드화된다.:

```
<methodCall>
  <methodName>sample.sumAndDifference</methodName>
  <params>
    <param><value><int>5</int></value></param>
    <param><value><int>3</int></value></param>
  </params>
</methodCall>
```

이것은 장황해 보이지만 쉽게 요약된다. 게다가 생각보다 빠르다. Rick Blair의 측정에 따르면, Hannes Wallnofer의 Java 환경에서 한 번의 XML-RPC 호출 왕복에 0.003초가 걸렸다.

2.2. 지원되는 데이타형

XML-RPC는 다음과 같은 데이타형을 지원한다:

`int`

부호 있는 32비트 정수형

`string`

NULL 바이트를 포함할 수 있는 아스키 문자열 (XML의 특성에 의해 몇몇의 XML-RPC 환경에서는 유니코드를 지원함)

`boolean`

참 또는 거짓

`double`

배정도 실수형 (환경에 따라 제한)

`dateTime.iso8601`

날짜와 시각. XML-RPC는 타임존의 사용을 금하기 때문에 거의 쓸모 없음

`base64`

임의의 길이를 가지는 저수준 이진 데이타. 통신 상에서는 Base64로 인코딩되며 상당히 유용한 데이타형 (어떤 환경에서는 0 바이트의 데이타는 수신하지 못하는 경우도 있음)

`array`

1차원 배열. 배열값은 어떠한 형도 가능함

`struct`

key-value의 쌍으로 이루어진 사전형. key는 string이어야 하며 value는 어떠한 형도 가능함

2.3. XML-RPC의 역사

XML-RPC는 그 이전의 두 프로토콜로부터 영향을 받았다. 첫째는 Dave Winer에 의해 디자인되고 예전에 DaveNet (http://davenet.userland.com/1998/02/27/rpcOverHttpViaXml)에 발표된 익명 RPC 프로토콜이다. (이 때문에 종종 XML-RPC 서버는 `/RPC2`에 설치된다.) 또하나, 더욱 주요하게는 SOAP 프로토콜의 초안에서 그 영향을 받았다.

XML-RPC의 역사 (http://www.xmlrpc.com/stories/storyReader$555)에 관한 더 자세한 문서로는 Dave Winer가 쓴 글이 있다. 이 문서는 XML-RPC와 SOAP의 관계에 대해서도 설명하고 있다.

3. XML-RPC과 다른 프로토콜

원격 프로시져를 호출하는 방법에 XML-RPC가 유일한 것은 아니다. CORBA, DCOM 그리고 SOAP 등의 다른 유명한 프로토콜도 있다. 이들 프로토콜은 제각각 장단점을 가지고 있다.

이 장의 의견은 순전히 개인적인 생각일 뿐이다. 그냥 참고삼아 읽어주기 바란다.

3.1. XML-RPC 대 CORBA

CORBA (http://www.corba.org/)는 분산, 객체지향 애플리케이션의 작성에 많이 사용되는 프로토콜이다. CORBA는 대체로 멀티-tier 엔터프라이즈 애플리케이션에 많이 사용되어져 왔으며, 최근에는 Gnome (http://www.gnome.org/) 프로젝트에서 애플리케이션 간의 내부적 통신을 위해 사용되고 있다.

CORBA는 많은 업체와 자유 소프트웨어 프로젝트에 의해 지원되고 있다. CORBA는 Java, C++ 그리고 다른 여러 언어에서 잘 실행된다. 또한 CORBA는 사용자로 하여금 쉽고 객체지향적인 API를 정의할 수 있도록, 훌륭한 *인터페이스 정의 언어*(interface definition language: IDL)를 지원한다.

하지만 CORBA는 아주 복잡하다. 배우는 데 많은 시간이 들고, 적용하는 데에도 상당한 노력이 필요하며, 클라이언트 구현이 꽤나 어렵다. 그래서 CORBA는 웹 애플리케이션보다는 엔터프라이즈 또는 데스크탑 애플리케이션에 더 적합하다.

3.2. XML-RPC 대 DCOM

DCOM (http://www.microsoft.com/com/tech/DCOM.asp) 은 CORBA에 대응하는 Microsoft의 프로토콜이다. DCOM은 사용자가 이미 COM 컴포넌트를 사용하고 있고, Microsoft 외의 다른 시스템을 고려하지 않는다면 아주 훌륭하다. 그렇지 않다면 그다지 쓸모 없다.

3.3. XML-RPC 대 SOAP

SOAP (http://www.w3.org/TR/SOAP/)는 XML-RPC와 매우 유사하다. SOAP 역시 HTTP와 XML 문서를 통한 일련화 프로시져 호출을 사용한다. 하지만 SOAP는 최초의 표준 명세에서 서로 달리 확장해 가는 것을 막을 길이 없어 보인다.

SOAP는 원래 UserLand와 DevelopMentor, Microsoft의 협력에 의해 만들어졌다. 최초의 공개 릴리즈는 XML-RPC를 기본으로 하고 네임스페이스와 긴 엘리먼트 이름을 사용할 수 있도록 하였다. 그러나 그 이후에 SOAP는 W3C working group으로 넘겨졌다.

그런데 W3C working group은 SOAP에 여러 잡다한 것을 추가시켜 버렸다. 이 글을 쓰고 있는 현재 SOAP는 XML Schemas, enumerations, struct와 array의 혼합, 그리고 사용자 정의 데이타형 등을 지원한다. 그와 동시에 플랫폼에 따라 몇가지 형태의 SOAP가 나타났다.

XML-RPC과 유사하지만 더 많은 기능을 지원하는 프로토콜을 찾는다면 SOAP를 살펴보라. :-)

4. XML-RPC의 공통 인터페이스

몇몇의 XML-RPC 서버는 내장 메쏘드를 제공하고 있다. 그것은 XML-RPC 자체의 부분은 아니고 직접 만들어 추가한 것이다.

4.1. Introspection: 서버 API 찾기

Edd Dumbill는 다음과 같은 메쏘드를 제안했다 (http://xmlrpc.usefulinc.com/doc/reserved.html).:

```
array system.listMethods ()
string system.methodHelp (string methodName)
array system.methodSignature (string methodName)
```

만약 서버가 위의 메쏘드를 지원한다면 사용자는 그것을 질의하고 결과를 출력할 수 있다.

```
import xmlrpclib
server = xmlrpclib.Server("http://xmlrpc-
c.sourceforge.net/api/sample.php")
for method in server.system.listMethods():
    print method
    print server.system.methodHelp(method)
    print
```

위의 메쏘드들은 현재 서버상(http://xmlrpc-c.sourceforge.net)에서 PHP, C 그리고 Microsoft .NET의 형태로 제공되고 있다. 최근에 UserLand Frontier가 부분적으로 introspection을 지원하도록 업데이트 되었다. Perl, Python과 Java의 Introspection 지원에 대해서는 XML-RPC Hacks (http://xmlrpc-c.sourceforge.net/hacks.php) 페이지를 참고하라. 다른 XML-RPC 서버에서도 마음대로 introspection을 지원해도 좋다(권장한다)!

다양한 클라이언트 툴(문서 생성기, 래퍼 생성기 등등)도 XML-RPC Hacks 페이지에서 찾을 수 있다.

4.2. Boxcarring: 다수의 질의를 동시에 보내기

만약 작은 함수를 호출하는 루틴을 많이 가지는 XML-RPC 클라이언트를 만든다면, 인터넷 backbone latency를 이용하여 질의/응답의 왕복 시간을 줄이는 방법을 찾을 수 있다. 몇몇의 서버는 다음의 함수를 사용하여 다수의 질의를 하나로 묶어 보낼 수 있도록 한다:

```
array system.multicall (array calls)
```

상세한 정보는 system.multicall RFC (http://www.xmlrpc.com/discuss/msgReader$1208)에서 찾을 수 있다.

이 메쏘드는 현재 C와 UserLand Frontier로 작성된 서버들이 지원하고 있다. Python과 Perl로 작성된 서버는 XML-RPC Hacks (http://xmlrpc-c.sourceforge.net/hacks.php) 페이지에 있는 코드를 사용할 수 있다.

5. 샘플 API: `sumAndDifference`

XML-RPC를 설명하기 위해 가능한 한 많은 언어로 다음의 API를 구현해 보았다.

```
struct sample.sumAndDifference (int x, int y)
```

이 함수는 인수로 두 정수를 받아서 두 원소를 가지는 XML-RPC `<struct>`를 돌려준다:

sum

두 정수의 합

difference

두 정수의 차

별로 유용하진 않으나, 괜찮은 예제라 할 수 있다. :-)

이 함수(그리고 다른 함수들도)는 `http://xmlrpc-c.sourceforge.net/api/sample.php`을 통해 사용할 수 있다. (이 URL은 웹브라우저로는 동작하지 않는다. XML-RPC 클라이언트가 필요하다.)

6. Perl로 XML-RPC 사용하기

Ken MacLeod는 Perl로 XML-RPC를 구현했다. 그의 웹사이트 (http://bitsko.slc.ut.us/~ken/xml-rpc/)나 CPAN (http://www.cpan.org/)을 통해 Frontier::RPC 모듈을 구할 수 있다.

Frontier::RPC를 설치하려면, 패키지를 내려받고 Perl의 표준 방식으로 컴파일하면 된다:

```
bash$ gunzip -c Frontier-RPC-0.07b1.tar.gz | tar xvf -
bash$ cd Frontier-RPC-0.07b1
bash$ perl Makefile.PL
bash$ make
bash$ make test
bash$ su -c 'make install'
```

(이 과정은 윈도우즈나 root 계정이 아닌 경우에는 조금 다를 수 있다. 자세한 것은 Perl 문서를 참고하라.)

6.1. Perl 클라이언트

다음의 프로그램은 Perl로 XML-RPC 서버를 호출하는 방법을 보여준다:

```
use Frontier::Client;

# Make an object to represent the XML-RPC server.
$server_url = 'http://xmlrpc-c.sourceforge.net/api/sample.php';
$server = Frontier::Client->new(url => $server_url);

# Call the remote server and get our result.
$result = $server->call('sample.sumAndDifference', 5, 3);
$sum = $result->{'sum'};
$difference = $result->{'difference'};

print "Sum: $sum, Difference: $difference\n";
```

6.2. 독립 Perl 서버

다음의 프로그램은 Perl로 XML-RPC 서버를 만드는 방법을 보여준다:

```
use Frontier::Daemon;

sub sumAndDifference {
    my ($x, $y) = @_;
    return {'sum' => $x + $y, 'difference' => $x - $y};
}
```

```
# Call me as http://localhost:8080/RPC2
$methods = {'sample.sumAndDifference' => \&sumAndDifference};
Frontier::Daemon->new(LocalPort => 8080, methods => $methods)
    or die "Couldn't start HTTP server: $!";
```

6.3. CGI 기반의 Perl 서버

`Frontier::RPC2`는 CGI 기반의 서버 지원 기능을 내장하고 있지는 않다. 하지만 필요한 대부분 것을 구현할 수 있다.

다음의 코드를 웹서버의 `cgi-bin` 디렉토리에 `sumAndDifference.cgi` 라는 이름으로 저장하라. (유닉스 시스템에서는 `chmod +x sumAndDifference.cgi`로 실행 가능하도록 만들어주어야 한다.)

```
#!/usr/bin/perl -w

use strict;
use Frontier::RPC2;

sub sumAndDifference {
    my ($x, $y) = @_;
    return {'sum' => $x + $y, 'difference' => $x - $y};
}

process_cgi_call({'sample.sumAndDifference' => \&sumAndDifference});


#==========================================================================
#  CGI Support
#==========================================================================
#  Simple CGI support for Frontier::RPC2. You can copy this into your CGI
#  scripts verbatim, or you can package it into a library.
#  (Based on xmlrpc_cgi.c by Eric Kidd <http://xmlrpc-
c.sourceforge.net/>.)

# Process a CGI call.
sub process_cgi_call ($) {
    my ($methods) = @_;

    # Get our CGI request information.
    my $method = $ENV{'REQUEST_METHOD'};
    my $type = $ENV{'CONTENT_TYPE'};
    my $length = $ENV{'CONTENT_LENGTH'};

    # Perform some sanity checks.
    http_error(405, "Method Not Allowed") unless $method eq "POST";
    http_error(400, "Bad Request") unless $type eq "text/xml";
```

```
    http_error(411, "Length Required") unless $length > 0;

    # Fetch our body.
    my $body;
    my $count = read STDIN, $body, $length;
    http_error(400, "Bad Request") unless $count == $length;

    # Serve our request.
    my $coder = Frontier::RPC2->new;
    send_xml($coder->serve($body, $methods));
}

# Send an HTTP error and exit.
sub http_error ($$) {
    my ($code, $message) = @_;
    print <<"EOD";
Status: $code $message
Content-type: text/html

<title>$code $message</title>
<h1>$code $message</h1>
<p>Unexpected error processing XML-RPC request.</p>
EOD
    exit 0;
}

# Send an XML document (but don't exit).
sub send_xml ($) {
    my ($xml_string) = @_;
    my $length = length($xml_string);
    print <<"EOD";
Status: 200 OK
Content-type: text/xml
Content-length: $length

EOD
    # We want precise control over whitespace here.
    print $xml_string;
}
```

이 유틸리티 루틴을 CGI 스크립트에 집어넣으면 된다.

7. Python으로 XML-RPC 사용하기

Fredrik Lundh는 훌륭한 XML-RPC library for Python (http://www.pythonware.com/products/xmlrpc/)을 제공했다.

설치하려면 최근 버전을 내려받아서 `*.py` 파일을 Python 코드가 있는 디렉토리에 두거나 Python 디렉토리에 설치하면 된다.

RedHat 6.2 사용자는 다음과 같이 하면 된다:

```
bash$ mkdir xmlrpclib-0.9.8
bash$ cd xmlrpclib-0.9.8
bash$ unzip ../xmlrpc-0.9.8-990621.zip
bash$ python
python> import xmlrpclib
python> import xmlrpcserver
python> Control-D
bash$ su -c 'cp *.py *.pyc /usr/lib/python1.5/'
```

두 `*.py` 파일을 컴파일하기 위해 `import`하는 방법을 사용했다. 다른 플랫폼의 사용자는 Python 문서를 참고하라.

다른 예제를 보기 위해서는 O'Reilly Network의 XML-RPC: It Works Both Ways (http://www.oreillynet.com/pub/a/python/2001/01/17/xmlrpcserver.html) 문서를 보라.

7.1. Python 클라이언트

다음의 프로그램은 Python으로 XML-RPC 서버를 호출하는 방법을 보여준다:

```
import xmlrpclib

# Create an object to represent our server.
server_url = 'http://xmlrpc-c.sourceforge.net/api/sample.php';
server = xmlrpclib.Server(server_url);

# Call the server and get our result.
result = server.sample.sumAndDifference(5, 3)
print "Sum:", result['sum']
print "Difference:", result['difference']
```

8. C/C++로 XML-RPC 사용하기

C/C++을 위한 XML-RPC는 xmlrpc-c 웹사이트 (http://xmlrpc-c.sourceforge.net) 에서 구할 수 있다.

관련 파일들은 RPM 형태로 구할 수도 있고, 소스를 내려받아 컴파일할 수도 있다.

8.1. C 클라이언트

다음의 코드를 `getSumAndDifference.c`의 이름으로 저장하라:

```
#include <stdio.h>
#include <xmlrpc.h>
#include <xmlrpc_client.h>

#define NAME       "XML-RPC getSumAndDifference C Client"
#define VERSION    "0.1"
#define SERVER_URL "http://xmlrpc-c.sourceforge.net/api/sample.php"

void die_if_fault_occurred (xmlrpc_env *env)
{
    /* Check our error-handling environment for an XML-RPC fault. */
    if (env->fault_occurred) {
        fprintf(stderr, "XML-RPC Fault: %s (%d)\n",
                env->fault_string, env->fault_code);
        exit(1);
    }
}

int main (int argc, char** argv)
{
    xmlrpc_env env;
    xmlrpc_value *result;
    xmlrpc_int32 sum, difference;

    /* Start up our XML-RPC client library. */
    xmlrpc_client_init(XMLRPC_CLIENT_NO_FLAGS, NAME, VERSION);
    xmlrpc_env_init(&env);

    /* Call our XML-RPC server. */
    result = xmlrpc_client_call(&env, SERVER_URL,
                                "sample.sumAndDifference", "(ii)",
                                (xmlrpc_int32) 5,
                                (xmlrpc_int32) 3);
    die_if_fault_occurred(&env);

    /* Parse our result value. */
    xmlrpc_parse_value(&env, result, "{s:i,s:i,*}",
                       "sum", &sum,
                       "difference", &difference);
    die_if_fault_occurred(&env);

    /* Print out our sum and difference. */
    printf("Sum: %d, Difference: %d\n", (int) sum, (int) difference);

    /* Dispose of our result value. */
    xmlrpc_DECREF(result);

    /* Shutdown our XML-RPC client library. */
    xmlrpc_env_clean(&env);
```

```
    xmlrpc_client_cleanup();

    return 0;
}
```

컴파일은 다음과 같이 한다:

```
bash$ CLIENT_CFLAGS=`xmlrpc-c-config libwww-client --cflags`
bash$ CLIENT_LIBS=`xmlrpc-c-config libwww-client --libs`
bash$ gcc $CLIENT_CFLAGS -
o getSumAndDifference getSumAndDifference.c $CLIENT_LIBS
```

각자의 시스템에 맞도록 위의 **gcc**를 바꿔주면 된다.

8.2. C++ 클라이언트

다음의 코드를 `getSumAndDifference2.cc`의 이름으로 저장하라:

```
#include <iostream.h>
#include <XmlRpcCpp.h>

#define NAME       "XML-RPC getSumAndDifference C++ Client"
#define VERSION    "0.1"
#define SERVER_URL "http://xmlrpc-c.sourceforge.net/api/sample.php"

static void get_sum_and_difference () {

    // Build our parameter array.
    XmlRpcValue param_array = XmlRpcValue::makeArray();
    param_array.arrayAppendItem(XmlRpcValue::makeInt(5));
    param_array.arrayAppendItem(XmlRpcValue::makeInt(3));

    // Create an object to resprent the server, and make our call.
    XmlRpcClient server (SERVER_URL);
    XmlRpcValue re-
sult = server.call("sample.sumAndDifference", param_array);

    // Extract the sum and difference from our struct.
    XmlRpcValue::int32 sum = result.structGetValue("sum").getInt();
    XmlRpc-
Value::int32 diff = result.structGetValue("difference").getInt();

    cout << "Sum: " << sum << ", Difference: " << diff << endl;
}

int main (int argc, char **argv) {

    // Start up our client library.
```

```
    XmlRpcClient::Initialize(NAME, VERSION);

    // Call our client routine, and watch out for faults.
    try {
        get_sum_and_difference();
    } catch (XmlRpcFault& fault) {
        cerr << argv[0] << ": XML-RPC fault #" << fault.getFaultCode()
             << ": " << fault.getFaultString() << endl;
        XmlRpcClient::Terminate();
        exit(1);
    }

    // Shut down our client library.
    XmlRpcClient::Terminate();
    return 0;
}
```

컴파일은 다음과 같이 한다:

```
bash$ CLIENT_CFLAGS=`xmlrpc-c-config c++ libwww-client --cflags`
bash$ CLIENT_LIBS=`xmlrpc-c-config c++ libwww-client --libs`
bash$ c++ $CLIENT_CFLAGS -
o getSumAndDifference2 getSumAndDifference2.cc $CLIENT_LIBS
```

구식 C++ 컴파일러로는 제대로 컴파일이 되지 않을 수도 있다.

8.3. Proxy Class를 사용하는 C++ 클라이언트

만약 XML-RPC 서버가 Introspection API를 지원한다면 자동으로 C++ proxy class를 만들 수 있다. Proxy class를 만들기 위해서는 다음의 명령을 입력하고 그 결과를 파일로 저장하라:

```
bash$ xml-rpc-api2cpp \
> http://xmlrpc-c.sourceforge.net/api/sample.php sample SampleProxy
```

위의 명령은 `SampleProxy`라는 이름의 proxy class를 만든다. SampleProxy는 `sample`로 시작하는 모든 메쏘드의 래퍼(wrapper)를 가지고 있다. 물론 원한다면 수정할 수 있다.

8.4. CGI 기반의 서버

다음의 코드를 `sumAndDifference.c`의 이름으로 저장하라:

```
#include <xmlrpc.h>
#include <xmlrpc_cgi.h>

xmlrpc_value *
```

```
sumAndDifference (xmlrpc_env *env, xml-
rpc_value *param_array, void *user_data)
{
    xmlrpc_int32 x, y;

    /* Parse our argument array. */
    xmlrpc_parse_value(env, param_array, "(ii)", &x, &y);
    if (env->fault_occurred)
        return NULL;

    /* Return our result. */
    return xmlrpc_build_value(env, "{s:i,s:i}",
                              "sum", x + y,
                              "difference", x - y);
}

int main (int argc, char **argv)
{
    /* Set up our CGI library. */
    xmlrpc_cgi_init(XMLRPC_CGI_NO_FLAGS);

    /* Install our only method (with a method signa-
ture and a help string). */
    xmlrpc_cgi_add_method_w_doc("sample.sumAndDifference",
                                &sumAndDifference, NULL,
                                "S:ii", "Add and subtract two integers.");

    /* Call the appropriate method. */
    xmlrpc_cgi_process_call();

    /* Clean up our CGI library. */
    xmlrpc_cgi_cleanup();
}
```

컴파일은 다음과 같이 한다:

```
bash$ CGI_CFLAGS=`xmlrpc-c-config cgi-server --cflags`
bash$ CGI_LIBS=`xmlrpc-c-config cgi-server --libs`
bash$ gcc $CGI_CFLAGS -o sumAndDifference.cgi sumAndDifference.c $CGI_LIBS
```

컴파일이 되었으면 `sumAndDifference.cgi` 파일을 웹서버의 `cgi-bin` 디렉토리에 복사하면 된다.

9. Java로 XML-RPC 사용하기

Hannes Walln膨秤er는 훌륭한 Java용 XML-RPC 라이브러리 (http://helma.at/hannes/xmlrpc/)를 제공했다.

설치하기 위해서는 파일을 받은 후 압축을 풀고 `*.jar` 파일을 CLASSPATH에 추가해야 한다. 유닉스 시스템에서는 다음과 같이 한다:

```
bash$ unzip xmlrpc-java.zip
bash$ cd xmlrpc-java/lib
bash$ CLASSPATH=`pwd`/openxml-1.2.jar:`pwd`/xmlrpc.jar:$CLASSPATH
```

9.1. Java 클라이언트

다음의 프로그램을 `JavaClient.java`의 이름으로 저장하라.

```
import java.util.Vector;
import java.util.Hashtable;
import helma.xmlrpc.*;

public class JavaClient {

    // The location of our server.
    private final static String server_url =
        "http://xmlrpc-c.sourceforge.net/api/sample.php";

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

            // Create an object to represent our server.
            XmlRpcClient server = new XmlRpcClient(server_url);

            // Build our parameter list.
            Vector params = new Vector();
            params.addElement(new Integer(5));
            params.addElement(new Integer(3));

            // Call the server, and get our result.
            Hashtable result =
                (Hashtable) server.execute("sample.sumAndDifference", params);
            int sum = ((Integer) result.get("sum")).intValue();
            int difference = ((Inte-
ger) result.get("difference")).intValue();

            // Print out our result.
            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());
        }
    }
}
```

9.2. 독립 Java 서버

다음의 프로그램을 `JavaServer.java`의 이름으로 저장하라.

```
import java.util.Hashtable;
import helma.xmlrpc.*;

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;
    }

    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());
        }
    }
}
```

10. PHP로 XML-RPC 사용하기

Edd Dumbill는 PHP용 XML-RPC 모듈을 만들었다. 모듈은 UsefulInc XML-RPC website (http://xmlrpc.usefulinc.com/)에서 구할 수 있다.

파일을 내려받아 압축을 푼 다음, `xmlrpc.inc`와 `xmlrpcs.inc` 의 두 파일을 PHP 스크립트와 같은 디렉토리에 복사하면 된다.

10.1. PHP 클라이언트

다음의 스크립트는 웹페이지에 XML-RPC 호출 루틴을 추가하는 방법을 보여준다.

```
<html>
<head>
<title>XML-RPC PHP Demo</title>
</head>
<body>
<h1>XML-RPC PHP Demo</h1>

<?php
include 'xmlrpc.inc';

// Make an object to represent our server.
$server = new xmlrpc_client('/api/sample.php',
                            'xmlrpc-c.sourceforge.net', 80);

// Send a message to the server.
$message = new xmlrpcmsg('sample.sumAndDifference',
                         array(new xmlrpcval(5, 'int'),
                               new xmlrpcval(3, 'int')));
$result = $server->send($message);

// Process the response.
if (!$result) {
    print "<p>Could not connect to HTTP server.</p>";
} elseif ($result->faultCode()) {
    print "<p>XML-RPC Fault #" . $result->faultCode() . ": " .
        $result->faultString();
} else {
    $struct = $result->value();
    $sumval = $struct->structmem('sum');
    $sum = $sumval->scalarval();
    $differenceval = $struct->structmem('difference');
    $difference = $differenceval->scalarval();
    print "<p>Sum: " . htmlentities($sum) .
        ", Difference: " . htmlentities($difference) . "</p>";
}
?>
```

```
</body></html>
```

웹서버에서 PHP 스크립트가 실행되지 않는다면 PHP 웹사이트 (http://www.php.net/)를 참조하라.

10.2. PHP 서버

다음의 스크립트는 PHP를 이용하여 XML-RPC 서버에 적용하는 방법을 보여준다.

```
<?php
include 'xmlrpc.inc';
include 'xmlrpcs.inc';

function sumAndDifference ($params) {

    // Parse our parameters.
    $xval = $params->getParam(0);
    $x = $xval->scalarval();
    $yval = $params->getParam(1);
    $y = $yval->scalarval();

    // Build our response.
    $struct = array('sum' => new xmlrpcval($x + $y, 'int'),
                    'difference' => new xmlrpcval($x - $y, 'int'));
    return new xmlrpcresp(new xmlrpcval($struct, 'struct'));
}

// Declare our signature and provide some documentation.
// (The PHP server supports remote introspection. Nifty!)
$sumAndDifference_sig = array(array('struct', 'int', 'int'));
$sumAndDifference_doc = 'Add and subtract two numbers';

new xmlrpc_server(array('sample.sumAndDifference' =>
                        array('function' => 'sumAndDifference',
                              'signature' => $sumAndDifference_sig,
                              'docstring' => $sumAndDifference_doc)));
?>
```

위의 스크립트를 웹서버에서 `http://localhost/path/sumAndDifference.php`와 같은 방식으로 실행시킬 수 있다.

11. Using XML-RPC with Microsoft .NET

Charles Cook이 현재 이 HOWTO에 대한 새로운 장을 쓰고 있다. 현재 온라인 초안 (http://www.cookcomputing.com/misc/howto.txt)을 볼 수 있다. 만약 이 페이지가 없어졌다면 리눅스 문서 프로젝트에서 XML-RPC HOWTO (http://www.linuxdoc.org/HOWTO/XML-RPC-HOWTO/index.html)(이 문서)의 최신버전을 살펴보라.

12. Ruby로 XML-RPC 사용하기

(이 장의 대부분은 **Michael Neumann**이 썼다.)

Ruby (http://www.ruby-lang.org/)는 객체지향적인 스크립트 언어다. 이미 일본에서는 프로그래밍의 주류를 이루고 있으며, 그 외의 곳으로도 확대되어 가고 있다.

Ruby로 XML-RPC를 사용하기 위해서는 먼저 Yoshida Masato의 xmlparser 모듈(James Clark의 expat 파서의 루비용 래퍼)을 먼저 설치해야 한다. 이 모듈은 Ruby Application Archive (http://www.ruby-lang.org/en/raa.html)에서 찾을 수 있다.

xmlrpc4r (http://www.s-direktnet.de/homepages/neumann/xmlrpc4r/index.html) 모듈을 다음과 같이 설치하라:

```
bash$ tar -xvzf xmlrpc4r-1_2.tar.gz
bash$ cd xmlrpc4r-1_2
bash$ su root -c "ruby install.rb"
```

12.1. Ruby 클라이언트

다음은 간단한 Ruby 클라이언트이다:

```
require "xmlrpc/client"

# Make an object to represent the XML-RPC server.
server = XMLRPC::Client.new( "xmlrpc-
c.sourceforge.net", "/api/sample.php")

# Call the remote server and get our result
result = server.call("sample.sumAndDifference", 5, 3)

sum = result["sum"]
difference = result["difference"]

puts "Sum: #{sum}, Difference: #{difference}"
```

12.2. Ruby 서버

다음은 간단한 Ruby 서버이다:

```
require "xmlrpc/server"

s = XMLRPC::CGIServer.new

s.add_hanlder("sample.sumAndDifference") do |a,b|
  { "sum" => a + b, "difference" => a - b }
end

s.serve
```

위의 코드는 다음과 같이 작성할 수도 있다:

```
require "xmlrpc/server"

s = XMLRPC::CGIServer.new

class MyHandler
  def sumAndDifference(a, b)
    { "sum" => a + b, "difference" => a - b }
  end
end

s.add_handler("sample", MyHandler.new)
s.serve
```

CGI 기반의 서버 대신 독립 서버로 실행하려면 두번째 줄을 다음과 같이 바꾸면 된다:

```
s = XMLRPC::Server.new(8080)
```

13. 간이 언어로 XML-RPC 사용하기

다음과 같은 비범용 간이 언어 환경에서도 XML-RPC를 사용할 수 있다.

13.1. K로 XML-RPC 사용하기

(이 장의 대부분은 **Christian Langreiter**가 썼다.)

K (http://www.kx.com/)는 재무(finance)와 데이타베이스 개발용 언어이다. K용 XML-RPC 모듈은 Kx 웹사이트 (http://www.langreiter.com/k/kxr)에서 구할 수 있다. 압축을 풀고 .k 프로그램이 있는 디렉토리에 파일을 두면 된다.

다음은 간단한 클라이언트이다:

```
\l kxr
server:("localhost";"/cgi-bin/sumAndDifference.cgi");
.kxr.c[server;"sumAndDifference";2 3]
```

또한 서버는 다음과 같다:

```
\l kxrsc
.kxr.sm: ,("sumAndDifference";{.((`sum;+/x);(`difference;-/x))})
```

다른 예제는 배포되는 파일 속에 들어있다.

14. XML-RPC 모듈을 내장한 애플리케이션

몇몇의 유명한 리눅스 애플리케이션은 XML-RPC 모듈을 자체적으로 지원하고 있다. 이 프로그램들은 다른 곳에서 이미 많이 언급되었으므로, 여기서는 링크만 걸어두고자 한다.

14.1. Zope

Zope의 XML-RPC 사용법에 대한 문서는 다음과 같은 곳에서 찾을 수 있다:

- XML-RPC Programming with Zope (http://www.byte.com/column/BYT19991021S0014) by Jon Udell
- Zope XML-RPC (http://linux.userland.com/stories/storyReader$18) at UserLand.Com

14.2. KDE 2.0

KDE 2.0은 Kurt Ganroth의 kxmlrpc daemon (http://kdecvs.stud.fh-heilbronn.de/cvsweb/kdebase/kxmlrpc/)을 포함한다. 이것은 KDE 애플리케이션이 XML-RPC를 사용할 수 있도록 해 주는 스크립트이다.

Python으로 만든 간단한 샘플 프로그램 (http://kdecvs.stud.fh-heilbronn.de/cvsweb/kdebase/kxmlrpc/test/testxmlrpc.py?rev=1.6&content-type=text/x-cvsweb-markup)도 있다. 이 프로그램은 kxmlrpc에 연결하여 KDE 주소록을 조작하고, KDE trader를 질의하는 방법을 보여준다.

만약 다른 문서나 코드가 있다면 15.3절을 보라. KDE 스크립트에 대한 더 많은 정보가 필요하다.

15. 이 문서에 대하여

이 문서는 리눅스 문서 프로젝트 (http://www.linuxdoc.org/)의 일부분이다. Dave Winer와 많은 XML-RPC 라이브러리의 관리자들에게 감사를 표한다.

15.1. 이 문서의 새 버젼

이 문서의 최신 버젼은 리눅스 문서 프로젝트 (http://www.linuxdoc.org/HOWTO/XML-RPC-HOWTO/index.html) 웹사이트에서 구할 수 있다.

다음과 같은 다른 형식의 문서도 지원한다: PDF (http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/pdf/XML-RPC-HOWTO.pdf), gzipped HTML (http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html/XML-RPC-HOWTO-html.tar.gz) and ASCII text (http://www.ibiblio.org/pub/Linux/docs/HOWTO/XML-RPC-HOWTO).

15.2. XML-RPC HOWTO 문서의 공헌자

Ruby에 대한 장은 Michael Neumann(neumann AT s-direktnet DOT de)이 도움을 주었다. K에 대한 장은 Christian Langreiter(c DOT langreiter AT synerge DOT at)가 도움을 주었다.

위의 공헌자들은 그들의 작성한 장의 첫머리에 이름이 붙는다.

15.3. 문서 보내기

만약 다른 언어나 환경에서 만든 클라이언트나 서버 예제가 있다면 그 설명서를 보내달라. 새로 등록하려면 다음과 같은 정보가 필요하다:

- XML-RPC 모듈이 사용된 URL
- 설치 방법
- 완전하게 실행되는 프로그램
- (컴파일해야 하는 것이라면) 컴파일 방법

xmlrpc-c-devel 메일링리스트 (http://xmlrpc-c.sourceforge.net/lists.php), 혹은 직접 Eric Kidd (mailto:eric.kidd@pobox.com)에게 예제를 메일로 보내달라.

Thank you for your help!