XML-RPC

Roberto Gómez Cárdenas
http://homepage.cem.itesm.mx/rogomez
rogomez@itesm.mx

Lámina 1 Roberto Gómez C.

Es necesario que dliente y servidor se entiendan entre ellos

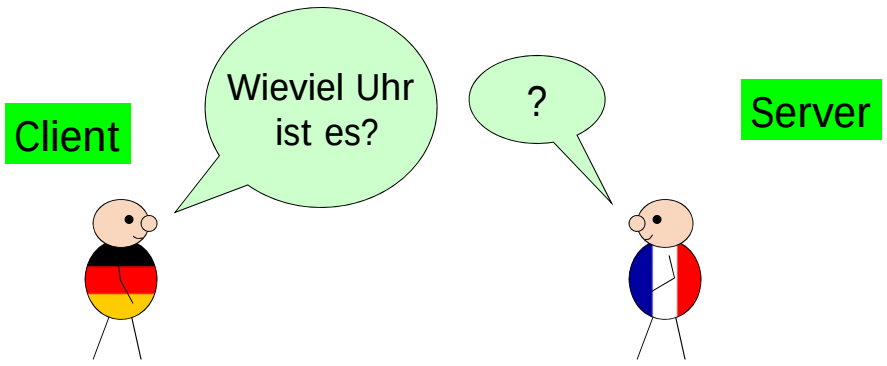

Lámina 2 Roberto Gómez C.

Cliente y servidor se ponen de acuerdo en un lenguaje en común.

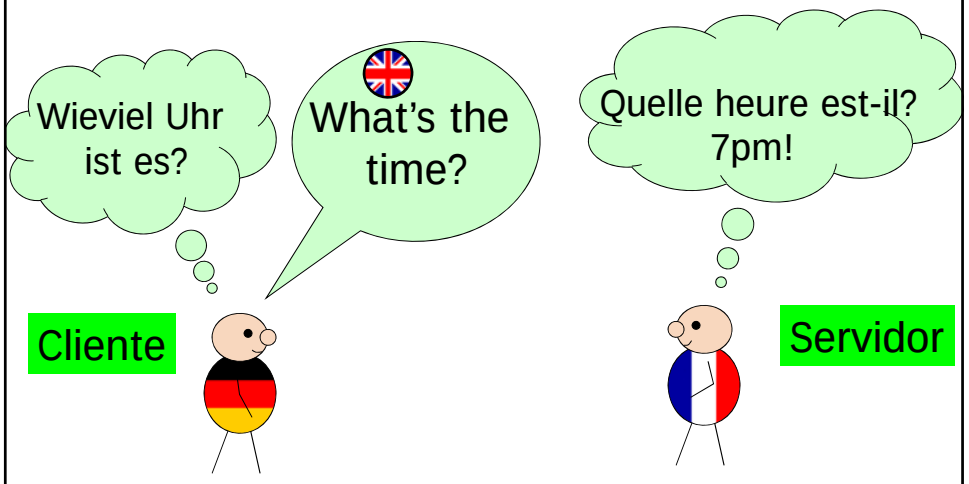

Lámina 3 Roberto Gómez C.

El cliente codifica la información en un lenguaje en común, el servidor decodifica la información.

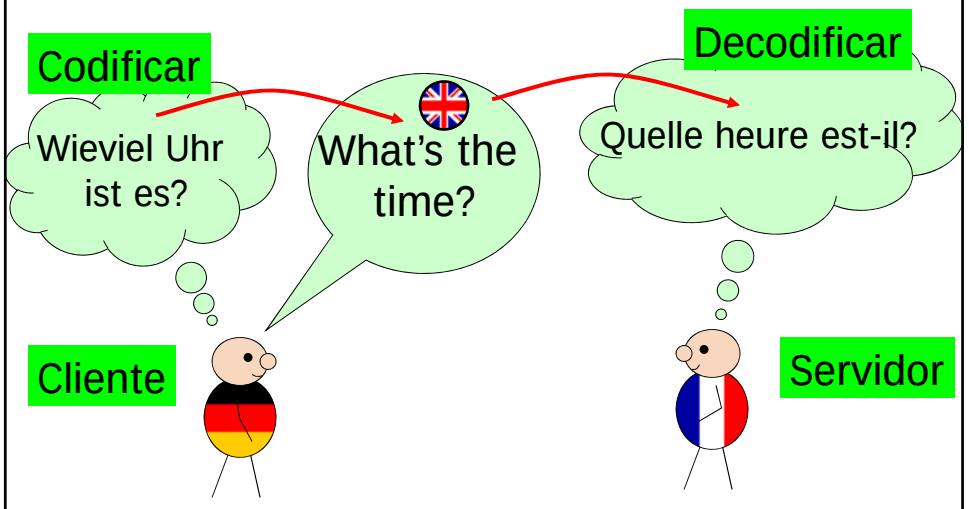

Lámina 4 Roberto Gómez C.

XML-RPC utiliza XML como un languaje común para transmitir datos

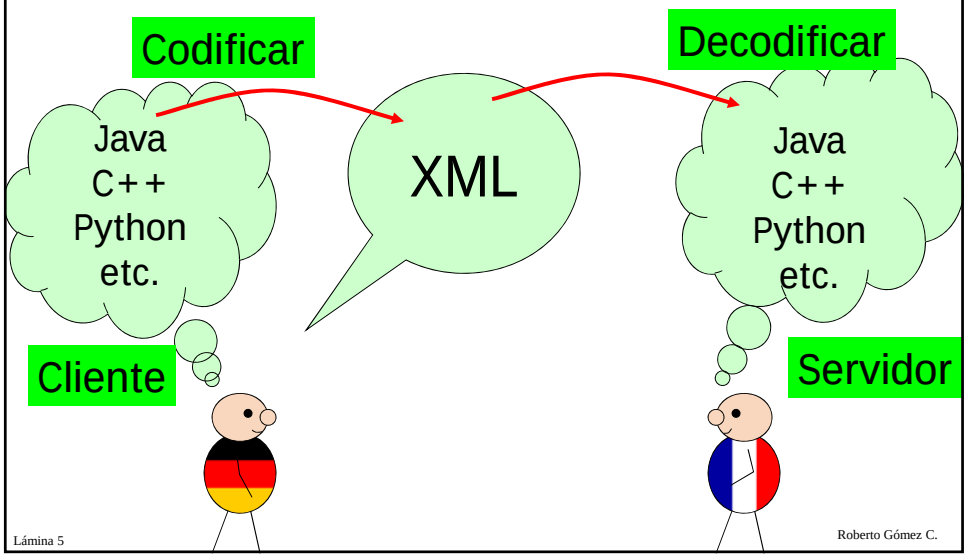

Lámina 5 Roberto Gómez C.

XML-RPC

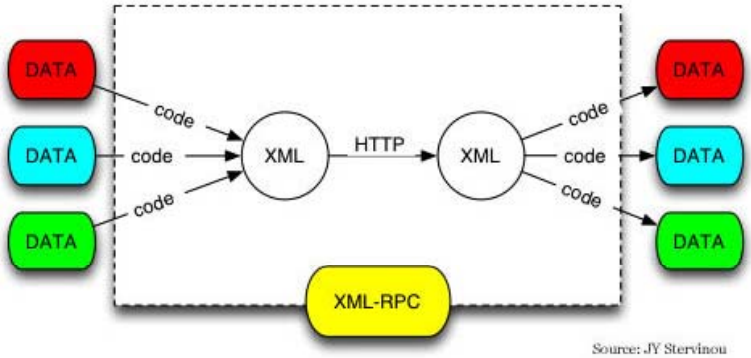

Lámina 6 Roberto Gómez C.

¿Qué es XML-RPC?

- Es una especificación y un conjunto de implementaciones que permiten que software corre en sistemas operativos diferentes, corriendo en ambientes distintos, llevar a cabo llamadas de procedimientos sobre internet.
- El llamado remoto usa HTTP como transporte y XML como codificación.
- Esta diseñado para ser tan simple como posible, permitiendo la transmisión de estructuras de datos complejas para ser transmitidas, procesadas y regresadas.

Lámina 7 Roberto Gómez C.

Arquitectura XML-RPC

- Arquitectura cliente-servidor.
- Cliente ejecutar un RPC en el servidor
- Servidor cuenta con tres componentes
  - El principal thread
  - El servidor XML-RPC (se hace cargo de los stubs)
  - Uno o más manejadores de RPC
- Servidor procesa cada RPC llamando el correspondiente manejador de RPC.

Lámina 8 Roberto Gómez C.

Arquitectura XML-RPC

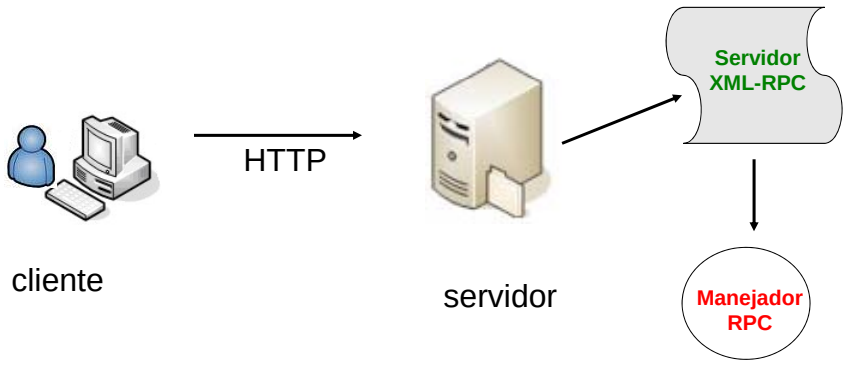

Lámina 9 Roberto Gómez C.

Tipos en XML-RPC

- Enteros con signo de 4 bytes
- Boolean
- String
- Double
- Fecha/Hora
- Binario (base 64)
- Arreglos de cualquiera de los tipos anteriores
- Estructuras conteniendo estos tipos.

Lámina 10 Roberto Gómez C.

Lenguajes que soportan XML-RPC

- Perl
- Python
- Java
- Frontier
- C/C++
- Lisp
- PHP
- Microsoft .NET
- Rebol
- Real Basic
- Tcl
- Delphi
- WebObjects
- Zope

Lámina 11 Roberto Gómez C.

Cliente XML-RPC

- Dos tipos de XML-RPCs
  - Sincronos
  - Asincronos
- ¿Cuál es la diferencia?
- ¿Cuáles son las ventajas y/o desventajas de cada uno de ellos?

Lámina 12 Roberto Gómez C.

XML-RPCs sincronos

- Cliente ejecuta el RPC en el servidor
  - Cliente se bloquea hasta que el RPC regresa
- Servidor regresa un objeto/tipo genérico.
- Cliente debe conocer el tipo a esperar
  - Llevar a cabo un “cast” del tipo genérico al tipo esperado.

Lámina 13 Roberto Gómez C.

XML-RPCs asíncronos

- El cliente debe definir un objeto de tipo call-back.
  - Método para manejar el éxito del RPC
  - Método para manejar la falla/error del RPC
- Cliente lleva a cabo el RPC asíncrono.
  - Simplemente lleva a cabo un nuevo thread para el RPC.
  - Regresa inmediatamente.
- Cuando el RPC ha terminado, uno de los métodos call-back es ejecutado.
  - El cast convierte el valor al tipo esperado.

Lámina 14 Roberto Gómez C.

XML-RPC y PHP

- Las clases de XML-RPC se pueden bajar de
  http://xmlrpc.usefulinc.com/php.html
- Las clases tienen la forma de código PHP, que puede ser incluido por otros scripts PHP para proporcionar funcionalidad XML-RPC.
- Los dos archivos principales en la distribución son:
  - xmlrpc.inc
    - Código que proporciona funcionalidad XML-RPC para el cliente.
  - xmlrpcs.inc
    - Cuando se usa junto con xmlrpc.inc, proporciona funcionalidad de servidor.
    - http://homepage.cem.itesm.mx/rogomez/tempo/xmlrpc-2.2.2.zip
- Necesarios una instrucción include( ).

Lámina 15 Roberto Gómez C.

Ejemplo invocación

- Invocación del método ex.getStateName
  - Toma un valor entero (int) y regresa una cadena de caracteres (str).
  - Dado un número regresa el correspon-diente nombre de estado.

```
include("xmlrpc.inc");

$thestate=32;

$c=new xmlrpc_client( "/RPC2", "betty.userland.com", 80 );
$f=new xmlrpcmsg( 'ex.getStateName',
            array(new xmlrpcval($thestate, "int" )));
$r=$c->send($f);
$v=$r->value( );

if (!$r->faultCode( )) {
   print "State number ". $thestate . " is " .
        $v->scalarval( ) . "<br />";
        print "<hr />I got this value back<br /><pre>" .
        htmlentities($r->serialize( )). "</pre><hr />\n";
} else {
   print "Fault: ";
   print "Code: " . $r->faultCode( ) .
   " Reason '" .$r->faultString( )."'<br />";
}
```

Lámina 16

La función xmlrpc_client

```
include("xmlrpc.inc");

$thestate=32;

$c=new xmlrpc_client( "/RPC2", "betty.userland.com", 80 );
$f=new xmlrpcmsg( 'ex.getStateName',
            array(new xmlrpcval($thestate, "int“ )));
$r=$c->send($f);
$v=$r->value(  );

if (!$r->faultCode(  )) {
    print "State number ". $thestate . " is " .
        $v->scalarval(  ) . "<br />";
        print "<hr />I got this value back<br /><pre>" .
        htmlentities($r->serialize(  )). "</pre><hr />\n";
} else {
    print "Fault: ";
    print "Code: " . $r->faultCode(  ) .
    " Reason '" .$r->faultString(  )."'<br />";
}
```

Constructor que prepara un objeto cliente que accede a un servidor en específico

Lámina 17 Roberto Gómez C.

El constructor xml_client( )

- Clase usada para representar un cliente de un servidor XML-RPC.
- Sintaxis

```
xmlrpc_clientnew xmlrpc_client(string $server_url);

xmlrpc_clientnew xmlrpc_client(string $server_path,
                               string $server_hostname,
                               int $server_port80,
                               string $transport'http');
```

- Ejemplos

```
$cliente1 = new xmlrpc_client("http://phpxmlrpc.sourceforge.net/server.php");

$cliente2 = new xmlrpc_client("/RPC2", "betty.userland.com", 80);
```

Lámina 18 Roberto Gómez C.

La función xmlrpcmsg y xmlrpcval

Constructor que prepara la invocación remota, usando el objeto xmlrpcval para empaquetar los parámetros

```
include("xmlrpc.inc");

$thestate=32;

$c=new xmlrpc_client( "/RPC2", "betty.userland.com", 80 );
$f=new xmlrpcmsg( 'ex.getStateName',
            array(new xmlrpcval($thestate, "int“ )));
$r=$c->send($f);
$v=$r->value(  );

if (!$r->faultCode(  )) {
    print "State number ". $thestate . " is " .
        $v->scalarval(  ) . "<br />";
        print "<hr />I got this value back<br /><pre>" .
        htmlentities($r->serialize(  )). "</pre><hr />\n";
} else {
    print "Fault: ";
    print "Code: " . $r->faultCode(  ) .
    " Reason '" .$r->faultString(  )."'<br />";
}
```

Lámina 19 Roberto Gómez C.

Constructor

- Permite la creación de un mensaje de petición XML-RPC.
- Necesario crear una instancia de la clase xmlrpcmsg
- Sintaxis

```
xmlrpcmsgnew xmlrpcmsg( string$methodName,
                        array$parameterArraynull);
```

  - methodName: nombre del metodo a invocar
  - parameterArray: arreglo de objetos xmlrpcval
- Ejemplo

```
$msg = new xmlrpcmsg("examples.getStateName",
                     array(new xmlrpcval(23, "int")));
```

Lámina 20 Roberto Gómez C.

La función send( )

```
include("xmlrpc.inc");

$thestate=32;

$c=new xmlrpc_client( "/RPC2", "betty.userland.com", 80 );
$f=new xmlrpcmsg( 'ex.getStateName',
            array(new xmlrpcval($thestate, "int“ )));
$r=$c->send($f);
$v=$r->value(  );

if (!$r->faultCode(  )) {
    print "State number ". $thestate . " is " .
        $v->scalarval(  ) . "<br />";
        print "<hr />I got this value back<br /><pre>" .
        htmlentities($r->serialize(  )). "</pre><hr />\n";
} else {
    print "Fault: ";
    print "Code: " . $r->faultCode(  ) .
    " Reason '" .$r->faultString(  )."'<br />";
}
```

Invoca al método remoto

Lámina 21

Roberto Gómez C.

La función value( ) y scalarval( )

```
include("xmlrpc.inc");

$thestate=32;

$c=new xmlrpc_client( "/RPC2", "betty.userland.com", 80 );
$f=new xmlrpcmsg( 'ex.getStateName',
            array(new xmlrpcval($thestate, "int“ )));
$r=$c->send($f);
$v=$r->value(  );

if (!$r->faultCode(  )) {
    print "State number ". $thestate . " is " .
        $v->scalarval(  ) . "<br />";
        print "<hr />I got this value back<br /><pre>" .
        htmlentities($r->serialize(  )). "</pre><hr />\n";
} else {
    print "Fault: ";
    print "Code: " . $r->faultCode(  ) .
    " Reason '" .$r->faultString(  )."'<br />";
}
```

El objerto $v encapsula el valor de regreso

scalarval( ) extrae el valor

Lámina 22

Roberto Gómez C.

La función faultcode( )

```
include("xmlrpc.inc");

$thestate=32;

$c=new xmlrpc_client( "/RPC2", "betty.userland.com", 80 );
$f=new xmlrpcmsg( 'ex.getStateName',
            array(new xmlrpcval($thestate, "int“ )));
$r=$c->send($f);
$v=$r->value(  );

if (!$r->faultCode(  )) {
    print "State number ". $thestate . " is " .
        $v->scalarval(  ) . "<br />";
        print "<hr />I got this value back<br /><pre>" .
        htmlentities($r->serialize(  )). "</pre><hr />\n";
} else {
    print "Fault: ";
    print "Code: " . $r->faultCode(  ) .
    " Reason '" .$r->faultString(  )."'<br />";
}
```

Usada para verificar condiciones de error

Lámina 23 Roberto Gómez C.

Mapeando datos entre XML-RPC y PHP

- Los tipos de datos que cuentan con un mapeo directo entre PHP y XML-RPC son int, string y double.
- Para preservar información el tipo de dato XML-RPC debe ser encapsulado en una clase, que contendrá el tipo en adición al valor.
  - PHP restringido con respecto a los arreglos ya que no puede diferenciar entre un arreglo convencional y un arreglo asociativo.
- Por esta razón XLM-RPC para PHP cuenta con su propio sistema de tipos.
  - Objetivo: un cliente escrito en un lenguaje tipeado débilmente se debe poder comunicar con un lenguaje tipeado fuertemente, y viceversa.
  - Por ejemplo un client e en PHP debe poder conectarse con un servidor escrito en C#

Lámina 24 Roberto Gómez C.

Poniendo Datos PHP en Etiquetas XML-RPC

- La clase xmlrpcval() proporciona un wraper type para un valor.
- Lo más simple es construir una cadena de caracteres.
- Por ejemplo, para construir un string con el valor “Hola Mundo”

```
$xstr=new xmlrpcval(“Hola Mundo”)
$xstr=new xmlrpcval(“Hola Mundo”, “string”)
```

Lámina 25 Roberto Gómez C.

Más ejemplo

- Construyendo un entero, un punto flotantes y un booleano.

```
$xint=new xmlrpcval(34, "int"); // un entero, 34
$xfloat=new xmlrpcval(23.435, "double"); // valor punto flotante, 23.435
$xbool=new xmlrpcval(1, "boolean");  // booleano
```

- Construir fechas/horas de tipo XML-RPC también es simple.

```
$xdate=new xmlrpcval("2000-11-18T20:30:05", "dateTime.iso8601");
```

- Se pueden construir las fechas/horas a partir de estampillas de Unix.

```
$xdate=new xmlrpcval(iso801_encode(time(  )));     // tiempo local
$xdate=new xmlrpcval(iso801_encode(time(  ), 1));
```

Lámina 26 Roberto Gómez C.

Objetos binarios y arreglos

- La construcción de un objeto binario (el de tipo base64) se hace a partir de:

```
$xbinary=new xmlrpcval($binaryData, "base64");
```

- Creación de un arreglo que contiene los números 1,2 y 3

```
$xarr=new xmlrpcval(array(
   new xmlrpcval(1, "int"),
   new xmlrpcval(2, "int"),
   new xmlrpcval(3, "int")), "array");
```

Lámina 27 Roberto Gómez C.

Una estructura

- Crear una estructura sigue el mismo patrón.
- Sin embargo en lugar de pasar un arreglo lineal al constructor xmlrpcval(), se pasa un arreglo asociativo en el que las llaves son los nombres de los miembros.
- Por ejemplo:

```
$xstruct=new xmlrpcval(array(
   "one"   => new xmlrpcval(1, "int"),
   "two"   => new xmlrpcval(2, "int"),
   "three" => new xmlrpcval(3, "int")), "struct");
```

Lámina 28 Roberto Gómez C.

Variables globales predefinidas para manejo de tipos

- $xmlrpcI4 = "i4";
- $xmlrpcInt = "int";
- $xmlrpcBoolean = "boolean";
- $xmlrpcDouble = "double";
- $xmlrpcString = "string";
- $xmlrpcDateTime = "dateTime.iso8601";
- $xmlrpcBase64 = "base64";
- $xmlrpcArray = "array";
- $xmlrpcStruct = "struct";

Lámina 29 Roberto Gómez C.

Pasando datos XML_RPC a PHP

- Cuando se recibe un resultado, este es entregado en formato XML-RPC.
- Necesario saber si el valor representa un escalar, un arreglo o una estructura.
- Si no se conoce el tipo se puede usar el método kindOf() para encontrar esto.
  - Este método regresa un valor tipo string que puede ser:
    - scalar
    - array
    - struct.

Lámina 30 Roberto Gómez C.

El método scalarval( )

- Si se requiere saber el tipo escalar, se puede usar el método scalartyp( ).
- Ejemplo:

```
// $xv es un valor de tipo XML-RPC
if ($xv->kindOf=="scalar") {
    print "I got a scalar, value " . $xv->scalarval( ) .
          ", of type " . $xv->scalartyp( ) . "<br />\n";
}
```

Lámina 31 Roberto Gómez C.

El método arraymem( )

- Si se trata de un arreglo XML-RPC, se puede acceder a sus miembros usando el método arraymem( ).
  - Toma un solo parámetro el índice del arreglo.
- El número de miembros en el arreglo puede ser determinado por el método arraysize( ).
- Recordar que cada miembro es un objeto xmlrpval.
- Ejemplo:

```
// $xv es un arreglo de enteros
for($a=0; $a<$xv->arraysize( ); $a++) {
    $v=$xv->arraymem($a);
    print "Got this number: " . $v->scalarval( ) . "<br />\n";
}
```

Lámina 32 Roberto Gómez C.

Manejo de estructuras

- Existen dos métodos, dependiendo si los miembros son conocidos con anterioridad o no.
- Si los miembros ya se conocen, se usa el método structmem( )
- Por ejemplo:

```
// $p es una estructura que representa datos personales
$age=$p->structmem("edad");
$name=$p->structmem("nombre");
print $name->scalarval() . " tiene " . $age->scalarval(  ) .
    " años.<br />\n";
```

Lámina 33 Roberto Gómez C.

Nota: Recorriendo elementos arreglos

- Un arreglo puede tener elementos de distintos tipos.
- No es necesario colocar el número de índice, ya que PHP lo asigna automáticamente para cada valor, comenzando siempre desde cero.
- Los índices, pueden no ser obligatoriamente consecutivos, ni tampoco comenzar de cero, ni tampoco ser un número.
- Un vector en PHP puede tener elementos en cualquier posición, por lo tanto, se puede cargar un vector con posiciones **no** consecutivas, sino en forma totalmente aleatoria.

Lámina 34 Roberto Gómez C.

Ejemplo arreglos en PHP (inicialización)

```
<Html>
<Title>  Ejemplo Arreglos en PHP  </Title>
<Body>
<?php
  // Inicializacion del arreglo:

    $Nombre[100] = "Javier";
    $Nombre[200] = "Cintia";
    $Nombre[150] = "Ricardo";
    $Nombre[350] = "Raúl";
    $Nombre[120] = "Guillermo";
```

Lámina 35 Roberto Gómez C.

Ejemplo arreglos PHP (impresión)

```
// Impresion del vector
  reset($Nombre);

  echo "<H2>"."Vector de Nombres";
  echo "<H3>"."<Hr>";
  while (list($i,$Valor)= each($Nombre) )
   {
     echo "Legajo: " . $i . " - ";
     echo "Nombre: " . $Valor;
     echo "<Br>";
    }
 ?>
</Body>
</Html>
```

La función reset(), llevar el índice al principio del arreglo.

La función list(), almacena en los parámetros ($i, $Valor), el índice y el valor devuelto por la función each(),.

La función each(), tiene como parámetro el vector puesto en juego

La función list(), avanza automáticamente el puntero al siguiente elemento del vector, y en caso de que el vector haya llegado al final, la función devuelve false.

Lámina 36 Roberto Gómez C.

Estructuras no conocidas

- Si los miembros no son conocidas se cuentan con dos métodos
  - structreset(); asigna el apuntador interno de la estructura a su primer campo
  - structeach(): regresa el índica actual y el valor asociado, avanzando el índice del arreglo.

Lámina 37 Roberto Gómez C.

Ejemplo

```
// $xs is un ejemplo de una estructura
$xs->structreset( );
while(list($key, $xval)=$xs->structeach( )) {
    print "Miembro '${key}' ";
    switch($xval->kindOf( )) {
    case "scalar":
        print “es de tipo escalar: " .
          $xval->scalartyp( ) . "<br />\n";
        break;
    default:
        print “es de tipo complejo: " .
          $xval->kindOf( ) . "<br />\n";
    }
}
```

Lámina 38 Roberto Gómez C.

El método xmlrpc_encode( )

- Se cuenta con una función que facilita la conversión de tipos PHP a XML-RPC
- Algunos ejemplos
  - Enteros

```
$xa=new xmlrpcval(23, "int");
$xa=xmlrpc_encode(23);
```

  - Dobles

```
$xb=new xmlrpcval(56.234, "double");
$xb=xmlrpc_encode(56.234);
```

  - Cadena de caracteres

```
$xb=new xmlrpcval(56.234, "double");
$xb=xmlrpc_encode(56.234);
```

Lámina 39 Roberto Gómez C.

Método xmlrpc_encode() y estructuras

- Mejor uso de xmlrpc_encode() es en estructuras.

```
// verbose construction
$person=new xmlrpcval(array(
        "name"   => new xmlrpcval("Fred Smith"),
        "age"    => new xmlrpcval(45, "int"),
        "height" => new xmlrpcval(1.84, "double")), "struct");

// shortcut construction
$person=xmlrpc_encode(array(
        "name"   => "Fred Smith",
        "age"    => 45,
        "height" => 1.84));
```

Lámina 40 Roberto Gómez C.

El método xmlrpc-decode( )

- Utilizado para pasar de XML-RPC a PHP.
- Ejemplo de uso para extraer el contenido de una estructura.

```
// $xv es una estructura xmlrpcval
$s=xmlrpc_decode($xv);
reset($s);
while(list($key, $val)=each($s)) {
    print "${key} is ${val}<br />\n";
}
```

Lámina 41 Roberto Gómez C.

Invocación de métodos

- Preparando un objeto cliente
  - Inicializar un objeto xmlrpc_client( )

    ```
    $client=new xmlrpc_client($server_path, $server_name, $server_port);
    ```
  - Ejemplo

    ```
    $client=new xmlrpc_client("/demo/server.php", "xmlrpc.usefulinc.com", 80);
    ```
- El llamar al constructor solo inicializa al objeto cliente, no lleva a cabo ninguna comunicación.
- Dos métodos pueden ser de utilidad
  - setDebug()

    ```
    $client->setDebug(1);
    ```
  - setCredentials( )

    ```
    $client->setCredentials($username, $password);
    ```

Lámina 42 Roberto Gómez C.

Preparando invocación método y sus parámetros

- Una vez el objeto cliente creado, se le debe pasar un objeto de mensaje, que se enviará al servidor.
- Mensaje encapsula el nombre del método a ser invocado, junto con sus parámetros. El objeto usado para este propósito es xmlrpcmsg( )

```
$msg=new xmlrpcmsg($methodName, array($p1, $p2 ... ));
```

  - Nombre método debe ser un string
  - Cada miembro del arreglo debe ser un objeto xmlrpcval
  - Es posible no contar con parámetros

```
$msg=new xmlrpcmsg($methodName);
```

  - Parámetros subsecuentes se pueden agregar con el método addParam( ) que toma un objeto xmlrpcval( ) como argumento

Lámina 43 Roberto Gómez C.

Depurando

- Con propósito de depuración se puede desear ver la codificación XML de la llamada del método XML-RPC.
- Ejemplo:

```
$msg=new xmlrpcmsg("examples.getStateName",
                   array(new xmlrpcval(23, "int")));
print "<pre>" . htmlentities($msg->serialize(  )) . "</pre>";
```

- La salida del método serialize( ), mostrando codificación XML es:

```
<?xml version="1.0"?>
<methodCall>
<methodName>examples.getStateName</methodName>
<params>
<param>
<value><int>23</int></value>
</param>
</params>
</methodCall>
```

Lámina 44 Roberto Gómez C.

Invocando el método

- Una vez que el objeto xmlrpcmsg ha sido creado, este puede ser pasado al método send() para enviárselo al servidor.
- Método send( ) puede ser invocado de dos formas.
  - La forma más simple es pasar el mensaje como un parámetro, el cliente se bloqueará hasta que se reciba una respuesta de objeto xmlrpcresp( ).

```
$result=$client->send($msg);
```

  - Se puede definir un timeout, y pasar el tiempo en segundo como segundo parámetro.

```
$result=$client->send($msg, 60); // timeout de 60 segundos
```

Lámina 45 Roberto Gómez C.

Verificando el resultado

- Enviar una mensaje XML-RPC a un servidor remoto puede dar como resultado:
  - Un error de bajo nivel de E/S
  - Una respuesta XML-RPC válida.
  - Una condición de error XML-RPC.
- Si ocurre un error, la información sobre este se puede acceder vía las variables errstr y errno.

Lámina 46 Roberto Gómez C.

Ejemplo manejo errores bajo nivel

- Errores bajo nivel: no se puede resolver el nombre del servidor o no se puede conectar a una máquina remota.

```
$result=$client->send($msg);
if (!$result) { // ningun error de bajo nivel ocurrio
   print “Aydua! Ha ocurrido un error de bajo nivel. Error # " .
        $client->errno . ": " . $client->errstr . "<br />\n";
   exit(0); // abortando
}
```

- Las variables $client->errno y $client->errstr normalmente corresponden a errores de bajo nivel definido por librerías de bajo nivel desde PHP.
- Estos errores corresponden a los representados por la variables errno de Unix.

Lámina 47 Roberto Gómez C.

Recuperación valor de regreso

- Para recuperar el valor, se cuenta con un método value( ), que regresa un objeto xmlrpcval()
- Ejemplo

```
$result=$client->send($msg);
if ($result) { // ningun error de bajo nivel ocurrio
   if ($result->value(  )) { // ningún error ocurrio
      $val=xmlrpc_decode($result->value(  ));
      print "Got this result: ${val}<br />\n";
   } else {
      // manejo del error de XML-RPC
   }
} else {
   // manejo del error de bajo nivel
}
```

Lámina 48 Roberto Gómez C.

Manejando condiciones de error

- Si valore de regreso es NULL una condición de error ocurrió.
- Estos son regresados al usuario con un número y una explicación legible por el usuario.
  - Metodos faultCode( ) y faultString( )

```
$result=$client->send($msg);
if ($result) {
    if (!$result->value(  )) { // se produjo una condición de error
        print “Ocurrio una condicion de error, numero código: " .
            $result->faultCode(  ) . ", explanation: " .
            $result->faultString(  ) . "<br />\n";
    }
}
```

Lámina 49 Roberto Gómez C.

Construyendo servidores XML-RPC en PHP

- El servidor XML-RPC es solo otro script PHP.
- El servicio se ofrece a través del puerto 80.
  - Se puede cambiar lo anterior en la configuración del servidor web.
- La tarea más importantes en definir procedimientos del servicio web,
- La librería PHP va más allá de los lenguajes permitiendo que el servidor sea auto-documentado.
- A través de una facilidad conocida como introspección, cualquier cliente puede solicita al servidor PHP la lista de procedimientos remotos implementados.

Lámina 50 Roberto Gómez C.

Métodos en PHP

- Un método toma la siguiente forma:

```
function myMethod($xmlrpcMessage) {
   $errorHappened=0;
   $param0=$xmlrpcMessage->getParam(0);
   $param1=$xmlrpcMessage->getParam(1);

   // .. Calculo del servicio

   if ($errorHappened) {
      $r=new xmlrpcresp(0, $errno, $errmsg);
   } else {
      $r=new xmlrpcresp(new xmlrpcval("Valor Regreso"));
   }
   return $r;
}
```

Objeto xmlrpcmsg como único parámetro

Método getNumParams( ) regresa el número total de parámetros

Parámetros son accedidos como objetos xmlrpval usando el método getParam( )

Lámina 51 Roberto Gómez C.

Regresando el resultado.

- Los valores son regresados a través de un objeto xmlrpcresp.
- El método debe regresar exactamente uno de estos objetos,
- Dos formas de construir un objeto xmlrpcresp:
  - new xmlrpcresp(0, $ errno, $ errmsg)
    - Contiene 0 como su primer argumento, indicando que el valor regresado es una cóndición de error.
    - El segundo argumento es el número de error.
    - El tercer argumento es la explicación del error.
  - new xmlrpcresp($ xmlrpcValue)
    - Usada para regresar un valor de un cálculo exitoso.
    - Un XML-RPC solo puede regresar un solo valor, que debe ser un objeto xmlrpcval.

Lámina 52 Roberto Gómez C.

¿Y el código del servicio?

- El código del servicio debe ir entre las líneas de código que acceden los parámetros y las líneas que crean el valor de regreso.
- Recordar que si un error ocurre, esto debe resultar en un xmlrcpresp regresado al cliente.
  - Usar rutinas como die( ) o warn( ) provocará que no se regrese nada al cliente.
- Errores de sintaxis u otros en el código del servidor provocará un payload malformado que será enviado al cliente.
- Activar la parte de depuración del lado del cliente para ver el error de PHP.

Lámina 53 Roberto Gómez C.

Ejemplo servidor (1/2)

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

  function aritmetica ($params) {

     // Recorriendo los parametros.
     $xval = $params->getParam(0);
     $x = $xval->scalarval();
     $yval = $params->getParam(1);
     $y = $yval->scalarval();

     // Haciendo el calculo
    $struct = array('suma' => new xmlrpcval($x + $y, 'int'),
                    'resta' => new xmlrpcval($x - $y, 'int'));

     // Regresando el resultado.
     return new xmlrpcresp(new xmlrpcval($struct, 'struct'));
  }
```

Lámina 54 Roberto Gómez C.

Ejemplo (2/2)

```
// Declarar el servicio y proporcionar documentación.
// El servidor PHP soporta introspección remota.

$aritmetica_sig = array(array('struct', 'int', 'int'));
$aritmetica_doc = 'Sumar y substraer dos numeros';

new xmlrpc_server(  array('server.aritmetica' =>
                          array('function' => 'aritmetica',
                          'signature' => $aritmetica_sig,
                          'docstring' => $aritmetica_doc)));
?>
```

Lámina 55 Roberto Gómez C.

Ejemplo cliente (1/2)

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

<?php
  include 'xmlrpc/lib/xmlrpc.inc';

  // Obteniendo un  objeto de manipulación de la llamada
  $server = new xmlrpc_client('/xmlrpc/server.php','localhost',80);

  // Solicitando el servicio
  $message = new xmlrpcmsg('server.aritmetica',
                                  array(new xmlrpcval(5, 'int'),
                                  new xmlrpcval(3, 'int')));

  // Recibiendo el resultado
$result = $server->send($message);
```

Lámina 56 Roberto Gómez C.

Ejemplo cliente (2/2)

```
// Procesando la respuesta
 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('suma');
      $sum = $sumval->scalarval();
      $differenceval = $struct->structmem('resta');
      $difference = $differenceval->scalarval();
      print "<p>Suma: " . htmlentities($sum) .
           ", Resta:" . htmlentities($difference) . "</p>";
  }
?>

</body>
</html>
```

Lámina 57 Roberto Gómez C.

XML-RPC

Roberto Gómez Cárdenas
http://homepage.cem.itesm.mx/rogomez
rogomez@itesm.mx

Lámina 58 Roberto Gómez C.