Tarea 9 Servicios web con SOAP

Enunciado

Vamos a seguir utilizando para esta tarea la base de datos correspondiente a la tienda web.

En primer lugar deberás utilizar el guión de comandos SQL para crear una base de datos de nombre “tarea6”, similar a la que usamos en los ejercicios anteriores. Para acceder se usan las credenciales habituales: usuario “dwes” y contraseña “abc123.”.

Guión de comandos SQL (0.02 MB)

A continuación utilizarás PHP5 SOAP para crear un servicio web con cuatro funciones que expongan información de la base de datos de la tienda online. De momento no deberás utilizar WSDL. Las funciones son las siguientes:

- `getPVP`. Esta función recibirá como parámetro el código de un producto, y devolverá el PVP correspondiente al mismo.
- `getStock`. Esta función recibirá dos parámetros: el código de un producto y el código de una tienda. Devolverá el stock existente en dicha tienda del producto.
- `getFamilias`. No recibe parámetros y devuelve un array con los códigos de todas las familias existentes.
- `getProductosFamilia`. Recibe como parámetro el código de una familia y devuelve un array con los códigos de todos los productos de esa familia.

El guión PHP que ejecute el servicio ha de llamarse `servicio.php`. Para comprobar la correcta ejecución del servicio, programa también un cliente con nombre `cliente.php` que realice una llamada a cada una de las funciones programadas y muestre el resultado obtenido.

Una vez finalizada la parte anterior, crea un nuevo servicio en un guión con nombre `serviciow.php`, idéntico al anterior, y coméntalo adecuadamente para obtener un WSDL del mismo utilizando la herramienta `WSDLDocument`. Publica el documento de descripción obtenido, `serviciow.wsdl`, en el servicio.

Partiendo de este nuevo servicio y de su descripción, utiliza la herramienta `wsdl2php` para obtener una clase en PHP. Crea un nuevo cliente llamado `clientew.php` que se base en ésta clase para probar el nuevo servicio, mostrando los resultados obtenidos de forma similar a como hiciste en el caso anterior.

Criterios de puntuación. Total 10 puntos.

Se valorará con dos puntos la consecución de cada uno de los siguientes ítems:

- Programar correctamente las funciones que se publicarán como parte del servicio web.
- Crear de forma correcta `servicio.php`.
- Comentar adecuadamente `serviciow.php`, para posibilitar la utilización de `WSDLDocument`.

Se valorará con un punto la consecución de cada uno de los siguientes ítems:

- Crear de forma correcta `cliente.php`.
- Crear de forma correcta `clientew.php`
- Utilizar `WSDLDocument` para obtener el documento de descripción del servicio web y utilizarlo como parte del mismo.
- Utilizar `wsdl2php` para obtener una nueva clase a partir del servicio web.

Recursos necesarios para realizar la Tarea.

Ordenador con PHP, servidor web Apache y entorno de desarrollo NetBeans, correctamente instalado y configurado. Extensión de depuración Xdebug para PHP instalada y funcionando con NetBeans.

Consejos y recomendaciones.

Además del manual online de PHP, es necesario dar libre acceso a Internet para la búsqueda de información.

”http://es.wikieducator.org/ManuelRomero/DAW/dwes/serviciosWebSoap”

Desarrollo de la tarea

A partir de los métodos que queremos servir 'getPVP', 'getFamilias', 'getProductosFamilia' y 'getStock' generamos los ficheros necesarios:

Server.php

```php
<?php
require_once('DB.php');
require_once('Producto.php');

class Server {

  /**
   * Obtiene el PVP de un producto a partir de su código
   * @param string $codigo
   * @return float
   */
  public function getPVP($codigo) {
    $producto = DB::obtieneProducto($codigo);
```

```
    $precio = $producto->getPVP();
    return $precio;
  }

  /**
   * Devuelve un array con los códigos de todas las familias
   * @return string[]
   */
  public function getFamilias() {
    $familias = DB::obtieneFamilias();
    return $familias;
  }

  /**
   * Devuelve un array con los códigos de los productos de una familia
   * @return string[]
   */
  public function getProductosFamilia($familia) {
    $productos = DB::obtieneProductosFamilia($familia);
    return $productos;
  }

  /**
   * Devuelve el número de unidades que existen en una tienda de un producto
   * @param string $codigo
   * @param int $tienda
   * @return int $unidades
   */
  public function getStock($codigo, $tienda) {
    $unidades = DB::obtieneStock($codigo, $tienda);
    return $unidades;
  }
}
?>

?>
```

Producto.php

```
<?php

class Producto {
    protected $codigo;
    protected $nombre;
    protected $nombre_corto;
    protected $PVP;
    protected $descripcion;
    protected $familia;

    public function getcodigo() {return $this->codigo; }
    public function getnombre() {return $this->nombre; }
    public function getnombrecorto() {return $this->nombre_corto; }
    public function getPVP() {return $this->PVP; }
    public function getdescripcion() {return $this->descripcion; }
    public function getfamilia() {return $this->familia; }

    public function __construct($row) {
        $this->codigo = $row['cod'];
        $this->nombre = $row['nombre'];
        $this->nombre_corto = $row['nombre_corto'];
        $this->PVP = $row['PVP'];
        $this->descripcion = $row['descripcion'];
        $this->familia = $row['familia'];
    }
}
?>
```

DB.php

```php
<?php

require_once('Producto.php');

class DB {

  /**
   * Ejecuta una consulta de una sentencia sql pasada como parámetro
   */
  protected static function ejecutaConsulta($sql) {
    try {
      $opc = array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8");
      $dsn = "mysql:host=theasker.infenlaces.com;dbname=theasker_varios";
      $usuario = 'Theasker';
      $contrasena = 'Theasker';
      $dwes = new PDO($dsn, $usuario, $contrasena, $opc);
      $resultado = null;
      if (isset($dwes)) {
        $resultado = $dwes->query($sql);
      }
      return $resultado;
    } catch (PDOException $ex) {
      echo $ex->getCode() . ": " . $ex->getMessage();
    }
  }

  /**
   * Obtiene un objeto producto con todos los campos de un producto dado.
   */
  public static function obtieneProducto($codigo) {
    $sql = "SELECT cod, nombre_corto, nombre, PVP, descripcion, familia";
    $sql .= " FROM producto WHERE cod='" . $codigo . "'";
    $resultado = self::ejecutaConsulta($sql);
    $producto = null;
    if (isset($resultado)) {
      $row = $resultado->fetch();
      $producto = new Producto($row);
    }
    return $producto;
  }

  /**
   * Devuelve un array con los códigos de todas las familias
   */
  public static function obtieneFamilias(){
    $sql = "SELECT cod FROM familia";
    $resultado = self::ejecutaConsulta($sql);
    $familias = array();
    if ($resultado){
      $row = $resultado->fetch();
      while ($row != null){
        $familias[] = $row['cod'];
        $row = $resultado->fetch();
      }
    }
    return $familias;
  }

  /**
   * Devuelve un array con los códigos de los productos de una familia
   */
  public static function obtieneProductosFamilia($familia){
    $sql = "SELECT cod FROM producto WHERE familia = '".$familia."'";
    $resultado = self::ejecutaConsulta($sql);
    $codigosProd = array();
    if ($resultado){
      $row = $resultado->fetch();
      while ($row != null) {
        $codigosProd[] = $row['cod'];
        $row = $resultado->fetch();
```

```
      }
      return $codigosProd;
    }
  }

  /**
   * Devuelve el número de unidades que existen en una tienda de un producto
   */
  public static function obtieneStock($codigo, $tienda){
    $sql = "SELECT unidades FROM stock";
    $sql .= " WHERE producto = '".$codigo."' AND tienda = ".$tienda;
    $resultado = self::ejecutaConsulta($sql);
    $unidades = null;
    if ($resultado){
      $row = $resultado->fetch();
      $unidades = $row['unidades'];
    }
    return $unidades;
  }

}
?>
```

Parte 1 - Sin fichero WSDL

Crearemos un objeto de la case SoapServer para que haya un objeto en el servidor esperando a que haya una solicitud del cliente y atenderla.

servicio.php

```
<?php
require_once './Server.php';

//Sin WSDL -> uri es obligatorio
$uri = "http://localhost/www/dwes/php/tarea9_servicios_web/";
$server = new SoapServer(null, array('uri' => $uri));

$server->setClass('Server');
$server->handle();
?>
```

- Observa que creamos un objeto del tipo SoapServer
- Cargamos en ese objeto la clase que contiene los métodos del servicio
- Con handle generamos la respuesta del fichero html.

Usar el servicio

- Ahora ya tenemos el servicio y el objeto que esta escuchando esperando que alguien quiera usarlo
- Implementamos el fichero cliente
- En él instanciamos un objeto de la clase SoapClient como se ve en el fichero descrito a continuación

cliente.php

```
<?php
$url = "http://localhost/www/dwes/php/tarea9_servicios_web/servicio.php";
$uri = "http://localhost/www/dwes/php/tarea9_servicios_web/";
```

```
//Creamos un cliente para llamar a esa URL.
//Es obligatorio establecer el parámetro 'uri' al no tener WSDL ,
//igual que ocurría al instanciar el objeto SoapServer

$cliente = new SoapClient(null, array('location'=>$url,'uri'=>$uri));

//Ahora consumiremos todos los servicios de que disponemos
var_dump($cliente->getPVP('3DSNG'));
var_dump($cliente->getFamilias());
var_dump($cliente->getProductosFamilia('ORDENA'));
var_dump($cliente->getStock('OPTIOLS1100',1));
?>
```

Parte 2 - Con fichero WSDL

Ahora estamos en el caso de que esta interfaz de los métodos (nombre, parámetros y valor de retorno) no son conocidas por los que las quieren usar. Para ello las debemos publicar es un documento WSDL. Para generar este docuemento tenemos utilidades que automatzan (a mano es algo complicado, cuando menos tedioso).

Pasos a realizar:

1. Primero debemos comentar los métodos con las directivas o marcas usados por lenguejes que generan documentación como **javadoc** o **phpDocumentor**

```
...
  /**
   * Obtiene el PVP de un producto a partir de su código
   * @param string $codigo
   * @return float
   */
  public function getPVP($codigo) {
    $producto = DB::obtieneProducto($codigo);
    $precio = $producto->getPVP();
    return $precio;
  }
...
```

- Entre los tipos usaremos **int** para entero **string** para caracter **string[]** para un array o registro
- Una vez hecho esto procedemos a generar un fichero php para que usando WSDLDocument http://code.google.com/p/wsdldocument/ se nos genere el fichero correspondiente
- Como indican en los apuntes escribimos el fichero genera.php

genera.php

```
<?php

//serverW.php es el fichero descrito anteriormente,
// donde se implementan los métodos que se se ofrecen
require_once 'ServerW.php';
require_once '../WSDLDocument-0.6/src/WSDLDocument.php'; //script que generará el fichero xml

$url = "http://localhost/www/dwes/php/tarea9_servicios_web/Con_fichero_WSDL/servicioW.php";
$uri = "http://localhost/www/dwes/php/tarea9_servicios_web/Con_fichero_WSDL/";


//serviciow.php es el fichero que genera el objeto servidor soap
$accion = new WSDLDocument("ServerW", $url, $uri);

//header('Content-Type: text/xml');
```

```
echo $accion->saveXML(); //Genera el en  navegador el fichero xml que hay que revisar
/* LOS PARAMETROS DE WSDLDOCUMENT(   )
  1º: El nombre de la clase que gestionará las peticiones al servicio.
  2º: La URL en que se ofrece el servicio.
  3º: El espacio de nombres destino.
 * */
?>
```

pero como me da problemas, uso otro generador de archivos WSDL (https://code.google.com/p/php-wsdl-creator/) con el todo es más sencillo creando un fichero php con estás 2 líneas:

```
<?php
require_once './php-wsdl-2.3/class.phpwsdl.php';
PhpWsdl::RunQuickMode ( 'ServerW.php' ); // Archivo en el que están los métodos que queremos servir
?>
```

- Revisa la cabecera del xml por si tubieras que añadir algún nombre de dominio ns, fíjate en los ejemplos de los apuntes
- Coge el contenido que sale por la pantalla del navegador y pégala en un fichero. nombralo con extensión xml

Generando la clase a partir del wsdl wsdl2php

Ahora tenemos el xml que describe el servicio soap el fichero **`serviciow.wsdl.xml`** Nuestro objeto servidor deberá publicar el interfaz de los métodos que servimos:

servicioW.php

```
<?php
require_once './ServerW.php';

$xml = "http://localhost/www/dwes/php/tarea9_servicios_web/Con_fichero_WSDL/serviciow.wsdl.xml";
$server = new SoapServer($xml);

$server->setClass('ServerW');
$server->handle();
?>
```

Ahora descargamos la herramienta **wsdl2php** que nos generará el archivo php que contendrá los métodos que queremos servir según las especificaciones del fichero WSDL.

Lo descargamos de http://sourceforge.net/projects/wsdl2php/ y lo instalamos

```
sudo pear install wsdl2php-0.2.1-pear.tgz
```

aunque también lo podemos ejectutar sin instalarlo. Ahora generamos la clase a partir de la especificación del fichero wsdl y podemos usar en el cliente:

```
wsdl2php serviciow.wsdl.xml
```

y generamos un fichero. Si observamos al final del fichero wsdl

```
wsdl2php serviciow.wsdl.xml
```

o también con el fichero php

```
php wsdl2php.php serviciow.wsdl.xml
```

y generamos un fichero. Si observamos al final del fichero wsdl

```
...
    <wsdl:service name="ServerW">
        <wsdl:port name="ServerWSoap" binding="tns:ServerWSoap">
            <soap:address location=
"http://theasker.dyndns.org/www/dwes/php/tarea9_servicios_web/Con_fichero_WSDL/pruebaPHPWSDL.php"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
...
```

1. El nombre del fichero que genera es ServerW.php
2. Vemos el fichero que ha generado

ServerW.php

```
<?php

/**
 * ServerW class
 *
 *
 *
 * @author    {author}
 * @copyright {copyright}
 * @package   {package}
 */
class ServerW extends SoapClient {

  private static $classmap = array(
                                    );

  public function ServerW($wsdl = "serviciow.wsdl.xml", $options = array()) {
    foreach(self::$classmap as $key => $value) {
      if(!isset($options['classmap'][$key])) {
        $options['classmap'][$key] = $value;
      }
    }
    parent::__construct($wsdl, $options);
  }

  /**
   * Obtiene el PVP de un producto a partir de su codigo
   *
   * @param string $codigo
   * @return float
   */
  public function getPVP($codigo) {
    return $this->__soapCall('getPVP', array($codigo),       array(
            'uri' => 'http://localhost/www/dwes/php/tarea9_servicios_web/Con_fichero_WSDL/',
            'soapaction' => ''
           )
      );
  }

  /**
   * Devuelve un array con los codigos de todas las familias
   *
   * @param
   * @return UNKNOWN
   */
  public function getFamilias() {
    return $this->__soapCall('getFamilias', array(),       array(
```

```
            'uri' => 'http://localhost/www/dwes/php/tarea9_servicios_web/Con_fichero_WSDL/',
            'soapaction' => ''
          )
      );
  }

  /**
   * Devuelve un array con los ccodigos de los productos de una familia
   *
   * @param string $familia
   * @return UNKNOWN
   */
  public function getProductosFamilia($familia) {
    return $this->__soapCall('getProductosFamilia', array($familia),       array(
            'uri' => 'http://localhost/www/dwes/php/tarea9_servicios_web/Con_fichero_WSDL/',
            'soapaction' => ''
          )
      );
  }

  /**
   * Devuelve el nÃºmero de unidades que existen en una tienda de un producto
   *
   * @param string $codigo
   * @param int $tienda
   * @return int
   */
  public function getStock($codigo, $tienda) {
    return $this->__soapCall('getStock', array($codigo, $tienda),       array(
            'uri' => 'http://localhost/www/dwes/php/tarea9_servicios_web/Con_fichero_WSDL/',
            'soapaction' => ''
          )
      );
  }

}
?>
```

Usando las clases del servicio

- Ahora usamos la clase generada anteriormente
- Esta clase que extiende ***SoapCliente***
- En ella tenemos los métodos que el servidor ofrece como un servicio
- Lo ejecutamos y verificamos su funcionamiento.

```
<?php
 require_once 'ServerW.php';


//Instanciamos un objeto de la case ServerW
 //Esta clase se ha generado automáticamente por la herramienta wsdl2php
$cliente = new ServerW();

// Probamos a obtener el array con todas las familias
$familias = $cliente->getFamilias();
print_r($familias);
print "<br />";

// Probamos a obtener todos los productos de una familia
$productos = $cliente->getProductosFamilia("ORDENA");
print_r($productos);
print "<br />";
```

```php
// Probamos a obtener el precio de un producto
$pvp = $cliente->getPVP("KSTMSDHC8GB");
print("El PVP es ".$pvp);
print "<br />";

// Probamos a obtener el stock de un producto en una tienda
$unidades = $cliente->getStock("KSTMSDHC8GB", 3);
print("Existen ".$unidades." unidades");
?>
```