Desarrollo web en entorno servidor

Unidad 8: Programación de servicios web

ÍNDICE

INTRODUCCIÓN

A menudo necesitaremos utilizar **aplicaciones de terceros** para completar funcionalidades de nuestro código. Habitualmente son servicios que brindan terceras empresas y para utilizarlos debemos entendernos con ellos a nivel de comunicación.

Estas empresas deben **proveer datos** de una forma más o menos estandarizada para que puedan ser utilizados por múltiples clientes. Cuando hablamos de clientes nos referimos a **código escrito** con un lenguaje de programación.

Los **servicios web** resuelven este problema permitiendo que dos aplicaciones distintas, en máquinas distintas, con distinta **arquitectura de software** puedan comunicarse.

En esta unidad didáctica veremos **qué protocolos existen** y probaremos uno de ellos, creando y consumiendo un **servicio web** o **webservice**.

OBJETIVOS / CAPACIDADES

En esta unidad de aprendizaje, las capacidades que más se van a trabajar son:

- ✔ **Ajustar parámetros analizando la configuración para gestionar servidores de aplicaciones.**
- ✔ **Utilizar herramientas y lenguajes específicos, cumpliendo las especificaciones, para desarrollar e integrar componentes software en el entorno del servidor web.**
- ✔ **Evaluar servicios distribuidos ya desarrollados, verificando sus prestaciones y funcionalidad, para integrar servicios distribuidos en una aplicación web.**
- ✔ **Programar y realizar actividades para gestionar el mantenimiento de los recursos informáticos.**

PROYECTO DE LA UNIDAD

Antes de empezar a trabajar el contenido, te presentamos la **actividad** que está relacionada con esta unidad de aprendizaje. Se trata de un **caso práctico** basado en una **situación real** con la que te puedes encontrar en tu puesto de trabajo. Con esta actividad se evaluará la puesta en práctica de los **criterios de evaluación** vinculados al resultado de aprendizaje que se trabaja en esta unidad. Para realizarla deberás hacer lo siguiente: lee el enunciado que te presentamos a continuación, dirígete al área general del módulo profesional, concretamente a la actividad de evaluación que se encuentra dentro de esta unidad, allí encontrarás todos los detalles sobre fecha y forma de entrega, objetivos... A lo largo de la unidad irás adquiriendo los conocimientos necesarios para ir elaborando este proyecto.

Enunciado:

Consumiendo datos de empleados.

Siguiendo con las actividades de evaluación anteriores, crea un servicio web que sea capaz de responder el salario bruto de un empleado a partir de su identificador.

Puedes crear el cliente como desees, ya sea una consulta CLI o con una página web.

El servicio web deberá recoger la petición, acceder a la base de datos, y responder con el dato concreto, o en su defecto, con un mensaje de error si el empleado no existe.

1. ARQUITECTURAS DE PROGRAMACIÓN ORIENTADAS A SERVICIOS

La **arquitectura orientada a los servicios** (**SOA**, siglas del inglés Service Oriented Architecture) es un tipo de diseño de software que permite **reutilizar** sus elementos gracias a las interfaces de servicios que se comunican a través de una red con un lenguaje común.

➔ **Funcionamiento**.

Un **servicio** es una unidad autónoma de una o más funciones del software diseñada para realizar una **tarea específica**, como recuperar cierta información o ejecutar una operación. Contiene las integraciones de datos y código que se necesitan para llevar a cabo una función empresarial completa y diferenciada. Se puede acceder a él de forma remota e interactuar con él o actualizarlo de manera independiente.

TOMA NOTA

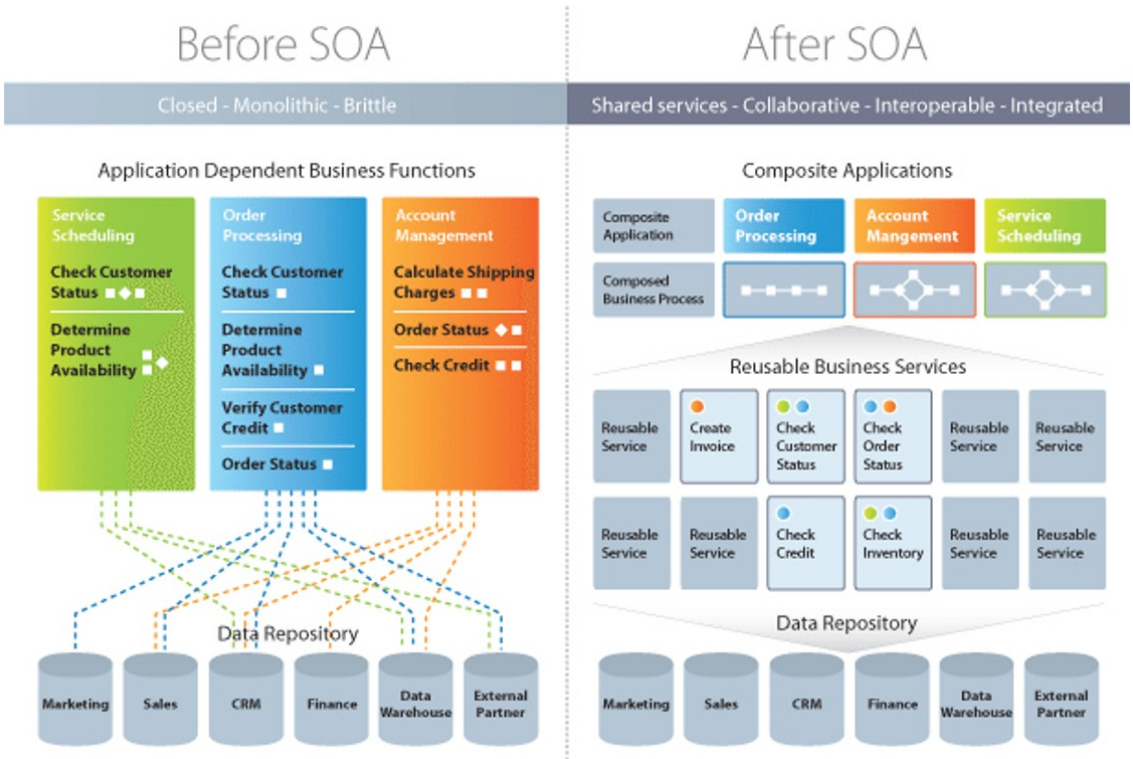
*En resumen, la SOA integra **pequeños componentes** de software que se implementan y se mantienen por separado, y permite que **se comuniquen entre sí y trabajen en conjunto** para formar aplicaciones de software en distintos sistemas.*

Si **SOA es la arquitectura**, es decir, una forma de diseñar software, **hará falta una implementación**. De mecanismos hay muchos, como veremos en el siguiente punto, pero, a fin de cuentas, todas necesitan una **API, Aplication Programing Interface**, o sea, una interfaz que exponga un componente para que otro componente se pueda comunicar con él.

➔ **Ejemplo práctico**.

Toda esa teoría puede parecer muy compleja. Pensemos en un **ejemplo práctico**: imaginemos que nosotros estamos trabajando en una **aplicación PHP** para mostrar el tiempo de una ciudad, y queremos recoger estos datos del servicio meteorológico estatal, que tiene una aplicación que es capaz de devolvernos esta información. Gracias a los servicios, no debemos conocer cómo está implementada esa aplicación, ni con qué se ha programado, ni qué sistema operativo la sostiene. Basta con sabernos comunicar con ella.

➔ A continuación, se muestra una imagen en la que verás una **comparación de una aplicación** compleja sin orientación a servicios y con orientación a servicios:



Comparación de una aplicación compleja sin orientación a servicios y con orientación a servicios. Fuente: https://medium.com/. Esta imagen se reproduce acogiéndose al derecho de cita o reseña (art. 32 LPI), y está excluida de la licencia por defecto de estos materiales.

Vídeo: ¿qué es SOA?

Visualiza el siguiente vídeo en el que verás qué es SOA.
https://www.youtube.com/embed/o_Br2vZ4uQY

2. MECANISMOS Y PROTOCOLOS IMPLICADOS. SOAP, WSDL. UDDI

Los **servicios web**, **WS** de las siglas en inglés Web Services, engloban varias tecnologías como XML, SOAP, WSDL, UDDI, etc. Estas tecnologías permiten construir **soluciones de programación** para mensajes específicos y para problemas de integración de aplicaciones.

Existen diversos **estándares** relacionados con los servicios web:

- **XML**: en inglés, eXtensible Markup Language (XML), es una herramienta independiente de software y hardware para almacenar y transportar datos.
- **HTTP**: el protocolo de transferencia de hipertexto (en inglés, Hypertext Transfer Protocol, abreviado HTTP) es el protocolo de comunicación que permite las transferencias de información a través de archivos (XHML, HTML, etc.) en la World Wide Web.
- **SOAP**: en inglés, Simple Object Access Protocol (SOAP), proporciona una forma de comunicación entre aplicaciones que se ejecutan en diferentes sistemas operativos, con diferentes tecnologías y lenguajes de programación.
- **REST**: en inglés Representational State Transfer (REST) describe cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc).
- **WSDL**: del inglés Web Services Description Language (WSDL), es un lenguaje basado en XML que utiliza unas reglas determinadas para generar el documento de descripción de un servicio web.
- **UDDI**: del inglés Universal Description, Discovery, and Integration (UDDI), define un modo de publicar y encontrar información sobre servicios web.

La idea principal es que los **servicios web trabajan sobre el protocolo HTTP**, de ahí su sombre. Un cliente hace una petición por el puerto 80 (HTTP) y si usa un servicio web, en vez de recibir una página web, obtendrá la información solicitada en un formato concreto según protocolo.

*Los **servicios web han evolucionado con el tiempo**, pasando del XML-RPC donde existía un protocolo, a los servicios web más modernos como SOAP y REST que operan bajo el paraguas HTTP.*

3. SOAP

La mejor manera de **comunicarse** entre aplicaciones es a través de **HTTP**, ya que este protocolo compatible con todos los navegadores y servidores de Internet. En este sentido, **SOAP** fue creado para lograr esto.

DESTACADO

***SOAP** proporciona una forma de **comunicación entre aplicaciones** que se ejecutan en diferentes sistemas operativos, con diferentes tecnologías y lenguajes de programación.*

Veamos las **claves** necesarias para conocer **SOAP**:

- **¿Qué es SOAP?** Un mensaje SOAP es un documento XML ordinario que contiene los siguientes elementos:
  - ✔ Un elemento de sobre que identifica el documento XML como un mensaje SOAP.
  - ✔ Un elemento de encabezado que contiene información de encabezado.
  - ✔ Un elemento de cuerpo que contiene información de llamada y respuesta.
  - ✔ Un elemento de falla que contiene errores e información de estado.

  Todos los elementos anteriores se declaran en el espacio de nombres predeterminado para el sobre SOAP y el espacio de nombres predeterminado para la codificación SOAP y los tipos de datos.

- Algunas reglas básicas sobre la **sintaxis** de SOAP son:
  - ✔ Un mensaje SOAP debe estar codificado usando XML.

- ✔ Un mensaje SOAP debe usar el espacio de nombres del sobre SOAP.
- ✔ Un mensaje SOAP no debe contener una referencia DTD.
- ✔ Un mensaje SOAP no debe contener instrucciones de procesamiento XML.

➔ Un **ejemplo** de un mensaje de petición en SOAP sería:

```
<?xml vesion="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body xmlns:m="http://www.ejemplo.com/stock">
<m:GetStockPrice>
<m:StockName>Apple</m:StockName>
</m:GetStockPrice>
</soap:Body>
</soap:Envelope>
```

Este ejemplo es una petición (*request*) para obtener el precio de mercado de la compañía Apple. Dentro del `soap:Body` se encuentra en elemento `GetStockPrice` que es específico para la aplicación. No es un elemento SOAP, y lleva el nombre que llevará la función en el servidor al que será enviado la *request*. `StockName` es también específico, y es un argumento de la función.

El **mensaje de respuesta** de la aplicación en SOAP podría ser:

```
<?xml verson="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body xmlns:m="http://www.ejemplo.com/stock">
<m:GetStockPriceresponse>
<m:Price>315.24</m:Price>
</m:GetStockPriceresponse>
</soap:Body>
</soap:Envelope>
```

Dentro del elemento `soap:Body` hay un elemento `GetStockPriceResponse` con un hijo que es `Precio` que contiene el valor devuelto, ambos específicos de la aplicación.

Cuestionario

Lee el enunciado e indica la opción correcta:

¿Qué significa SOA?

a. Service Oriented Architecture.

b. Service Open Application.

c. Simple Object Access.

Lee el enunciado e indica la opción correcta:

Lenguaje basado en XML que describe un documento para un servicio web:

a. WDSL.

b. SOAP.

c. REST.

Lee el enunciado e indica la opción correcta:

En la comunicación con SOAP se utiliza:

a. JSON.

b. CSV.

c. XML.

4. GENERACIÓN DE UN SERVICIO WEB

Vamos a generar un servicio web muy simple que se encargará de **calcular la letra de un DNI**. Es este caso construiremos el servicio en PHP y lo consumiremos también con PHP, pero recuerda que SOAP es agnóstico a las tecnologías usadas. En este sentido, este apartado escribiremos el código necesario para el archivo del servidor, es decir, para el propio servicio web. En la carpeta pública de tu servidor crea la ruta **webservice** y dentro de esta ruta crea el archivo **calcularLetra.php** con el siguiente script:

Código: archivo del servidor

```
<?php
// Instanciamos un nuevo servidor SOAP
$uri="http://localhost/webservice";
$server = new SoapServer(null,array('uri'=>$uri));
$server->addFunction("letra");
$server->handle();

// Función para obtener la letra
function letra($dni) {
    $resultado=substr("TRWAGMYFPDXBNJZSQVHLCKE",$dni%23,1);
    return $resultado;
}
?>
```

- **I**. Lo más interesante de este script es que hemos creado un objeto de tipo **SoapServer**. Este objeto funciona a modo no-WDSL, por eso el primer

parámetro es *null*, y seguidamente le pasamos la *uri* que va a ser el destino del espacio de nombres para el servidor.

- **II**. Después añadiremos la función **letra()** a nuestro *webservice*. Finalmente lanzamos el método **handle()** para gestionar las peticiones SOAP.
- **III**. La función para calcular la letra del DNI se basa en la fórmula donde se recoge el residuo de la división del número del DNI y el número 23, obteniendo un rango de valores del 0 al 22. El resultado es el valor de una letra a partir de la siguiente tabla:

RESTO	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
LETRA	T	R	W	A	G	M	Y	F	P	D	X	B	N	J	Z	S	Q	V	H	L	C	K	E

- **IV**. En siguientes capítulos escribiremos el código necesario para consumir este servicio web.

*¡Atención! Asegúrate de tener habilitada la extensión de SOAP en tu **fichero php.ini**. Busca en él la línea extension=soap y si está comentada, descoméntala y reinicia tu servidor web.*

Vídeo: creación de un Webservice

Visualiza el siguiente vídeo en el que verás cómo crear un Webservice.

5. DESCRIPCIÓN Y LOCALIZACIÓN DEL SERVICIO

A pesar que ya fue generado un archivo WSDL para nuestro *webservice* y este podrá ser consumido desde diversas plataformas y lenguajes, existe un último reto: **localizarlo**.

DESTACADO

*El **protocolo Universal Description, Discovery, and Integration (UDDI**) es un estándar OASIS aprobado y un miembro clave de la pila de servicios web. Define un método estándar para publicar y descubrir los componentes de software basados en red de una arquitectura orientada a servicios (SOA). UDDI es avanzado por el Comité Técnico de Especificación UDDI de OASIS.*

➔ ¿Cómo surgió UDDI?

Surgió con la intención de centralizar webservices comunes, así como ofrecer un depósito central donde se puede acudir a realizar búsquedas de webservices específicos, las metodologías que han sido descritas anteriormente solo permiten que el webservice sea descubierto siempre y cuando nos sea proporcionado el archivo WSDL.

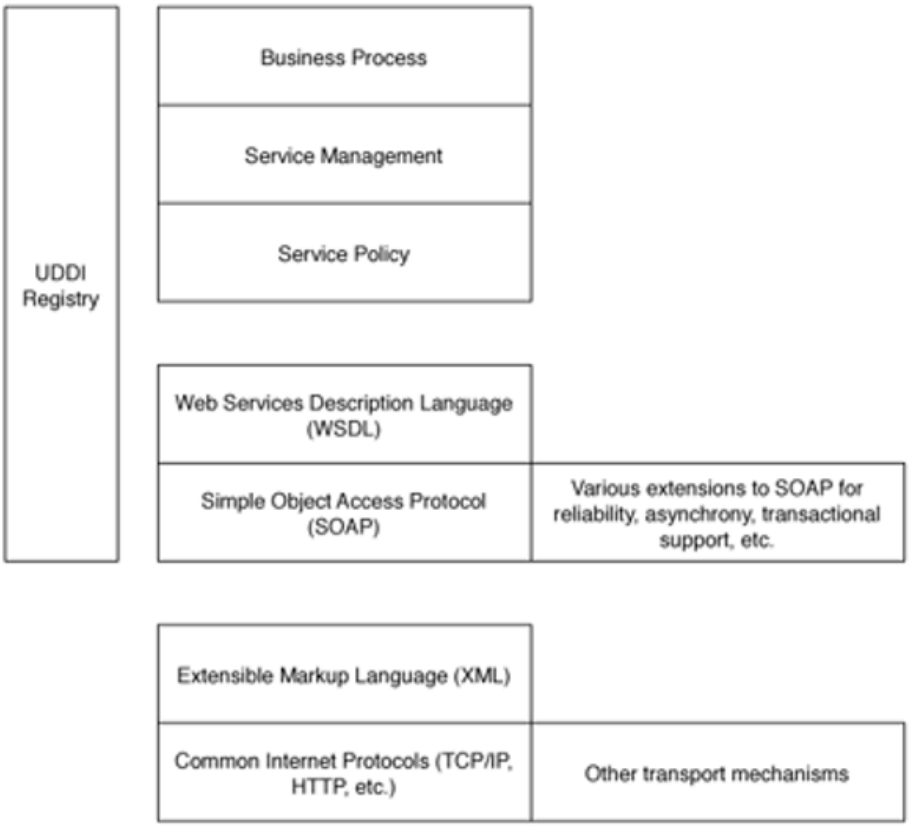
➔ Funcionamiento.

La información de UDDI se aloja en nodos de operador, empresas que se han comprometido a ejecutar un nodo público conforme a la especificación que rige el consorcio UDDI.org.

La especificación define un grupo de servicios web e interfaces programáticas para publicar, recuperar y administrar información sobre servicios. UDDI se

basa en varios otros estándares establecidos de la industria, incluidos HTTP, XML, XML Schema (XSD), SOAP y WSDL.

➔ La **relación conceptual entre UDDI y otros protocolos** en la pila de servicios web se ilustra en la siguiente imagen:



Relación UDDI con distintos servicios web. Fuente: http://uddi.xml.org/. Esta imagen se reproduce acogiéndose al derecho de cita o reseña (art. 32 LPI), y está excluida de la licencia por defecto de estos materiales.

Se pueden publicar los datos en un **nodo** y descubrirlos en otro tras la réplica. Actualmente, la réplica se produce cada 24 horas. En el futuro, este intervalo entre réplicas se reducirá, ya que habrá más **aplicaciones** que dependan de los datos de UDDI.

6. INTERFACE DE UN SERVICIO WEB

En este capítulo consumiremos el webservice que hemos creado para calcular la letra de un DNI. En la carpeta pública de tu servidor web crea el archivo **index.php** con el siguiente **script**:

Script

```
<?php
// Inicializamos variables
$error = "";
$resultado = "";
$dni = "";

// Iniciamos el cliente SOAP
// Escribimos la dirección donde se encuentra el servicio
$url = "http://localhost/webservice/calcularLetra.php";
$uri = "http://localhost/webservice/";
$cliente = new SoapClient(null, array('location' => $url, 'uri' => $uri));

// Ejecutamos las siguientes líneas al enviar el formulario
if (isset($_POST['enviar'])) {
    // Establecemos los parámetros de envío
    if (!empty($_POST['dni']) && (strlen($_POST['dni'])) >= 7) {
        $dni = $_POST['dni'];
         // Si los parámetros son correctos, llamamos a la función letra de calcularLetra.php
        $resultado = "La letra es: " . $cliente->letra($dni);
    } else {
            $error = "<strong>Error:</strong> Debes introducir un DNI correcto<br/><br/>Ej: 45678987";
    }
}
?>
```

```
<!DOCTYPE html>
<html>
    <head>
        <title>Calcular Letra DNI - Servicio Web + PHP + SOAP</title>
        <link rel="stylesheet" type="text/css" href="/estilo.css">
    </head>
<body>
    <h1>Obtener letra DNI</h1>
    <h2>Servicio Web + PHP + SOAP</h2>
    <form action="index.php" method="post">
    <?php
        print "<input type='text' name='dni' value='$dni'>";
             print "<input type='submit' name='enviar' value='Calcular
Letra'>";
        print "<p class='error'>$error</p>";
             print "<p style='font-size: 12pt;font-weight: bold;color:
#0066CC;'>$resultado</p>";
    ?>
    </form>
</body>
</html>
```

Al igual que hicimos con el servidor, ahora hemos creado un objeto SoapClient, de tipo no-WDSL donde el parámetro **url** es el recurso (**calcularLetra.php**) y el parámetro **uri** es el espacio de nombres.

Una vez comprobado que el usuario ha escrito un DNI correcto llamamos al *webservice* con la función `$cliente→letra($dni);`, aquí vemos la magia de los *webservices*. La función **letra()** sólo funcionará en este *webservice* en concreto, aunque la especificación el protocolo sea común para todos los *webservices*.

Actividad de aprendizaje 1: creación de un webservice

Actividad en parejas.

En la empresa donde trabajamos se está planteando migrar una web monolítica a microservicios creados con webservices. Para probar este nuevo paradigma, el jefe de programadores nos plantea un reto.

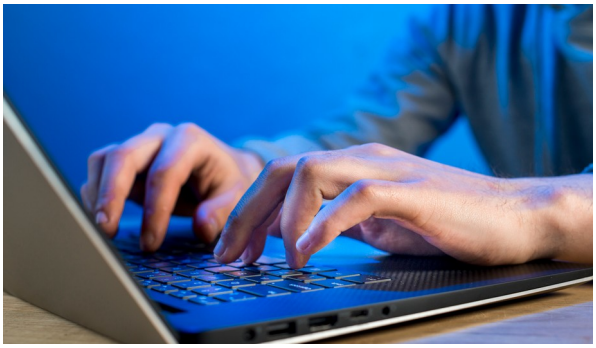
Uno de los miembros creará un formulario web que deberá conectarse a un servicio web pasándole una fecha, que será la fecha de nacimiento de un usuario.

El otro miembro deberá escribir el código del servicio web que calcule los años transcurridos entre la fecha pasada y la fecha actual y devolver la edad del usuario.

Entrega el resultado final en el espacio habilitado en el foro para ello.

7. UTILIZACIÓN DE UN SERVICIO WEB

Hasta ahora, hemos creado nosotros mismos un servicio web y posteriormente lo hemos utilizado. Pero el escenario más común es utilizar **servicios web de terceros**. Veamos ahora **cómo hacerlo**.

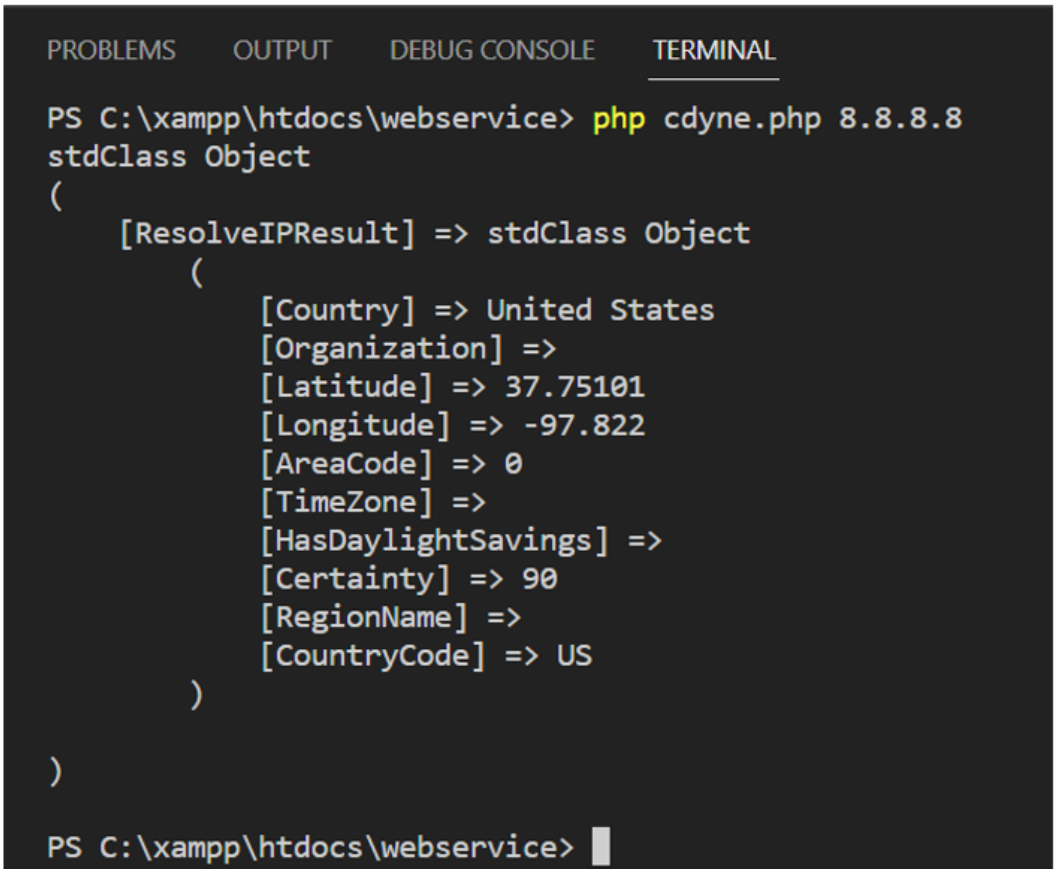


- **I**. Para este ejemplo, usaremos el WebService de **cdyne.com** para obtener información geográfica en base a la IP buscada.

- **II**. En cualquier directorio crearemos el archivo **cdyne.php** y escribiremos el siguiente script:

```
<?php

$url = "http://ws.cdyne.com/ip2geo/ip2geo.asmx?wsdl";

try {
 $client = new SoapClient($url, [ "trace" => 1 ] );
   $result   =   $client->ResolveIP(   [   "ipAddress"   =>   $argv[1],
"licenseKey" => "0" ] );

 print_r($result);
} catch ( SoapFault $e ) {
 echo $e->getMessage();
}

echo PHP_EOL;
?>
```

➔ **III**. Si ejecutamos este script desde CLI y probamos, por ejemplo, con la IP de los DNS de Google obtendremos una respuesta como la que vemos en la siguiente imagen:



Resultado de ejecutar la petición al webservice de cdyne

Vídeo: creación de un Webservice (II)

VÍDEO

Visualiza el siguiente vídeo en el que verás cómo crear otro Webservice más profesional.

Actividad de aprendizaje 2: consumo de webservice

En un entorno empresarial es conveniente saber los orígenes de las peticiones que nos llegan a nuestros servidores. A menudo, hay que hacer investigación por si hay alguna IP o una botnet que intente atacarnos. Para entrenar a localizar IPs, a partir de los ejemplos de la teoría sobre el consumo del servicio web de cdyne.com: crea un script que recupere los datos no a partir de CLI sino a partir de un formulario web y muestra el resultado en el navegador.

Entrega el resultado final en el espacio habilitado en el foro para ello.

Cuestionario

Lee el enunciado e indica la opción correcta:

Las siglas UDDI significan:

a. Universal Description, Discovery and Integration Directory.

b. Electronic Business XML Working Group.

c. United Data Directory Identificator.

Lee el enunciado e indica la opción correcta:

El objeto cliente de SOAP en PHP se instancia con la clase:

a. SoapC.

b. SoapClient.

c. ClientSoap.

Lee el enunciado e indica la opción correcta:

El objeto servidor de SOAP en PHP se instancia con la clase:

a. ServerSoap.

b. SoapServer.

c. SoapS.

RESUMEN

Con el aprendizaje del **uso de webservices** hemos incrementado exponencialmente el potencial para crear aplicaciones, ya sea siendo capaces de construir un **servicio web** para ofrecerlo a terceras empresas, o bien consumiendo un servicio de una tercera empresa para el desarrollo de nuestra aplicación.

A pesar que en esta unidad hemos implementado un servicio web con el **mecanismo SOAP** debemos recordar que hay otros estándares en la industria como las **RESTful API**.

Si entendemos el concepto de **webservice**, aunque sólo hayamos practicado con **SOAP**, nos será muy fácil, con unas pocas lecciones, ser capaces de escribir una aplicación **REST**.

RECURSOS PARA AMPLIAR

PÁGINAS WEB

- Consumir un webservice: *https://academy.leewayweb.com/como-consumir-un-webservice-soap-con-php/* [Consulta noviembre 2022].
- Crear un webservice: *https://www.raulprietofernandez.net/blog/programacion/como-crear-un-servicio-web-con-php-y-soap* [Consulta noviembre 2022].
- Espacio de nombres predeterminado para el sobre SOAP: *https://www.w3.org/2003/05/soap-envelope/* [Consulta noviembre 2022].
- Espacio de nombres predeterminado para la codificación SOAP y los tipos de datos: *https://www.w3.org/2003/05/soap-encoding* [Consulta noviembre 2022].

BIBLIOGRAFÍA

PÁGINAS WEB

- Cdyne: *https://cdyne.com* [Consulta noviembre 2022]
- Especificaciones del W3C sobre SOAP: *https://www.w3.org/TR/soap12-part1/* [Consulta noviembre 2022]
- Referencia oficial de PHP: *https://www.php.net/* [Consulta noviembre 2022]

GLOSARIO

- **HTTP**: siglas en inglés de Hypertext Transfer Protocol.
- **REST**: siglas en inglés de REpresentational State Transfer.
- **SOA**: siglas en inglés de Service Oriented Architecture, arquitectura orientada a los servicios.
- **SOAP**: siglas en inglés, Simple Object Access Protocol.
- **UDDI**: siglas en inglés de Universal Description, Discovery, and Integration.
- **WSDL**: siglas en inglés de Web Services Description Language.
- **XML**: siglas en inglés de eXtensible Markup Language.