Creado por:

Isabel Maniega

SOAP

SOAP (anteriormente conocido como Simple Object Access Protocol) es un protocolo ligero para el intercambio de información en entornos descentralizados y distribuidos. Los mensajes SOAP son las transmisiones de información de remitentes a destinatarios. Los mensajes SOAP se pueden combinar para crear patrones de petición/respuesta.

SOAP es independiente del transporte pero habitualmente se lleva a través de HTTP para ejecutarse con la infraestructura de Internet existente. SOAP habilita el enlace y la utilización de servicios Web descubiertos definiendo una vía de acceso de mensajes para direccionar mensajes.

SOAP es un protocolo basado en XML que define tres partes en todos los mensajes:

- Sobre. El sobre define una infraestructura para describir qué hay en un mensaje y cómo procesarlo. Un mensaje SOAP es un sobre que contiene cero o varias cabeceras y exactamente un cuerpo. El sobre es el elemento superior del documento XML que proporciona un contenedor para la información de control, la dirección de un mensaje y el mensaje en sí. Las cabeceras transportan información de control como por ejemplo atributos de calidad de servicio. El cuerpo contiene la identificación del mensaje y sus parámetros. Tanto las cabeceras como el cuerpo son elementos hijo del sobre.
- Reglas de codificación. El conjunto de reglas de codificación expresa instancias de tipos de datos definidos por la aplicación. Las normas de codificación definen un mecanismo de serialización que se puede utilizar para intercambiar instancias de tipos de datos definidos por la aplicación. SOAP define un esquema de tipo de datos independiente del lenguaje de programación basado en XSD y normas de codificación para todos los tipos de datos definidos de acuerdo a este modelo.
- Estilos de comunicación. Las comunicaciones pueden seguir el formato RPC (llamada a procedimiento remoto) u orientado a mensajes (Documento).

WSDL (Web Services Description Language)

WSDL (Web Services Description Language) es una especificación estándar para describir servicios basados en XML de red. Proporciona a los proveedores de servicios un modo sencillo de describir el formato básico de las peticiones a sus sistemas independientemente de la implementación del motor de ejecución subyacente.

WSDL define un formato XML para describir servicios de red como un conjunto de puntos finales que operan en mensajes que contienen información orientada a documentos u orientada a procedimientos. Primero se describen las operaciones y

mensajes de forma abstracta y luego se enlazan a un protocolo de red y formato de mensaje concreto para definir un punto final. Los puntos finales concretos relacionados se combinan en puntos finales abstractos (servicios). WSDL es ampliable para permitir la descripción de puntos finales y sus mensajes, independientemente de los formatos de mensaje o los protocolos de red que se utilicen para comunicarse. Esto significa que se definen las interfaces de forma abstracta con el esquema XML y luego se enlazan a representaciones concretas que son adecuadas para el protocolo.

WSDL permite a los proveedores de servicios especificar las siguientes características de un servicio Web:

- El nombre del servicio Web y la información de direccionamiento
- El protocolo y el estilo de codificación que se van a utilizar al acceder a las operaciones públicas del servicio Web
- La información de tipos, como las operaciones, los parámetros y los tipos de datos que componen la interfaz del servicio Web

Los documentos WSDL permiten a los desarrolladores exponer sus aplicaciones como servicios accesibles de red en Internet. Mediante UDDI y WSIL, otras aplicaciones pueden encontrar documentos WSDL y enlazarlos para ejecutar transacciones o realizar otros procesos de empresa.

WSDL Documents

Un documento WSDL describe un servicio web. Especifica la ubicación del servicio y los métodos del servicio mediante estos elementos principales:

Descripción del elemento:

`<types>` Define los tipos de datos (esquema XML) utilizados por el servicio web

`<message>` Define los elementos de datos para cada operación.

`<portType>` Describe las operaciones que se pueden realizar y los mensajes involucrados.

`<binding>` Define el protocolo y el formato de datos para cada tipo de puerto.

La estructura principal de un documento WSDL se ve así:

```
<definitions>

    <types>
      data type definitions........
    </types>

    <message>
      definition of the data being communicated....
    </message>

    <portType>
      set of operations......
```

```
    </portType>

    <binding>
      protocol and data format specification....
    </binding>

</definitions>
```

Módulo Zeep

Trabajar con servicios web basados en SOAP puede ser a veces una tarea que requiere mucho tiempo, ya que hay que escribir el XML completo para realizar solicitudes de API y luego analizar el XML de respuesta para obtener los resultados deseados. Eso es un dolor de cabeza, ¿no? Bueno, ahí es cuando Zeep entra en juego. Zeep es un módulo puramente Python. Lo mejor de Zeep es que no es necesario escribir el XML en absoluto. Solo hay que crear un diccionario con todos los datos de solicitud relevantes y Zeep creará el XML por usted.

Instalación con:

In [1]:
```
# pip install zeep
```

Ejemplo de como usar zeep

Tenemos un servicio SOAP activo para una calculadora (http://www.dneonline.com/calculator.asmx), podemos examinar el archivo WSDL de los serivios que contienen, para ello pondremos en el terminal el siguiente comando:

In [ ]:
```
# python -mzeep http://www.dneonline.com/calculator.asmx?WSDL
```

Nos devuelve la siguiente salida:

http://www.dneonline.com/calculator.asmx?WSDL

```
Prefixes:
     xsd: http://www.w3.org/2001/XMLSchema
     ns0: http://tempuri.org/

Global elements:
     ns0:Add(intA: xsd:int, intB: xsd:int)
     ns0:AddResponse(AddResult: xsd:int)
     ns0:Divide(intA: xsd:int, intB: xsd:int)
     ns0:DivideResponse(DivideResult: xsd:int)
     ns0:Multiply(intA: xsd:int, intB: xsd:int)
     ns0:MultiplyResponse(MultiplyResult: xsd:int)
     ns0:Subtract(intA: xsd:int, intB: xsd:int)
     ns0:SubtractResponse(SubtractResult: xsd:int)


Global types:
     xsd:anyType
     xsd:ENTITIES
```

```
        xsd:ENTITY
        xsd:ID
        xsd:IDREF
        xsd:IDREFS
        xsd:NCName
        xsd:NMTOKEN
        xsd:NMTOKENS
        xsd:NOTATION
        xsd:Name
        xsd:QName
        xsd:anySimpleType
        xsd:anyURI
        xsd:base64Binary
        xsd:boolean
        xsd:byte
        xsd:date
        xsd:dateTime
        xsd:decimal
        xsd:double
        xsd:duration
        xsd:float
        xsd:gDay
        xsd:gMonth
        xsd:gMonthDay
        xsd:gYear
        xsd:gYearMonth
        xsd:hexBinary
        xsd:int
        xsd:integer
        xsd:language
        xsd:long
        xsd:negativeInteger
        xsd:nonNegativeInteger
        xsd:nonPositiveInteger
        xsd:normalizedString
        xsd:positiveInteger
        xsd:short
        xsd:string
        xsd:time
        xsd:token
        xsd:unsignedByte
        xsd:unsignedInt
        xsd:unsignedLong
        xsd:unsignedShort

    Bindings:
        Soap11Binding: {http://tempuri.org/}CalculatorSoap
        Soap12Binding: {http://tempuri.org/}CalculatorSoap12

    Service: Calculator
        Port: CalculatorSoap (Soap11Binding:
    {http://tempuri.org/}CalculatorSoap)
            Operations:
               Add(intA: xsd:int, intB: xsd:int) ->
    AddResult: xsd:int
               Divide(intA: xsd:int, intB: xsd:int) ->
```

```
    DivideResult: xsd:int
                Multiply(intA: xsd:int, intB: xsd:int) ->
    MultiplyResult: xsd:int
                Subtract(intA: xsd:int, intB: xsd:int) ->
    SubtractResult: xsd:int

         Port: CalculatorSoap12 (Soap12Binding:
    {http://tempuri.org/}CalculatorSoap12)
             Operations:
                Add(intA: xsd:int, intB: xsd:int) ->
    AddResult: xsd:int
                Divide(intA: xsd:int, intB: xsd:int) ->
    DivideResult: xsd:int
                Multiply(intA: xsd:int, intB: xsd:int) ->
    MultiplyResult: xsd:int
                Subtract(intA: xsd:int, intB: xsd:int) ->
    SubtractResult: xsd:int
```

Nos fijamos en la parte de servicios que es lo que la aplicaicón es capaz de hacer (suma, división, multiplicación y resta), veremos como implementarlo usando el módulo zeep:

In [2]:
```
import os
import zeep

WSDL_URL="http://www.dneonline.com/calculator.asmx?WSDL"
soap = zeep.Client(wsdl=WSDL_URL)
dir(soap) # vemos los metodos disponibles, nos fijamos en el WSDL
```

Out[2]:
```
['__annotations__',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__enter__',
 '__eq__',
 '__exit__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 '_default_port_name',
 '_default_service',
 '_default_service_name',
 '_default_soapheaders',
 '_default_transport',
 '_get_port',
 '_get_service',
 'bind',
 'create_message',
 'create_service',
 'get_element',
 'get_type',
 'namespaces',
 'plugins',
 'service',
 'set_default_soapheaders',
 'set_ns_prefix',
 'settings',
 'transport',
 'type_factory',
 'wsdl',
 'wsse']
```

In [3]:
```
dir(soap.wsdl) # dentro de wsdl vemos que métodos tenemos
```

Out[3]:
```
['__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 '_add_definition',
 '_definitions',
 '_get_xml_document',
 'bindings',
 'dump',
 'load',
 'location',
 'messages',
 'port_types',
 'services',
 'settings',
 'transport',
 'types']
```

In [4]:
```
# accedemos a los servicios y pedimos las claves:

soap.wsdl.services.keys()
```

Out[4]:
```
odict_keys(['Calculator'])
```

In [5]:
```
# accedemos a las servicios y pedimos los elementos que lo componen:

soap.wsdl.services.items()
```

Out[5]:
```
odict_items([('Calculator', <Service(name='Calculator', ports=OrderedDic
t([('CalculatorSoap', <Port(name='CalculatorSoap', binding=<Soap11Bindin
g(name='{http://tempuri.org/}CalculatorSoap', port_type=<PortType(name
='{http://tempuri.org/}CalculatorSoap')>)>, {'address': 'http://www.dneo
nline.com/calculator.asmx'})>), ('CalculatorSoap12', <Port(name='Calcula
torSoap12', binding=<Soap12Binding(name='{http://tempuri.org/}Calculator
Soap12', port_type=<PortType(name='{http://tempuri.org/}CalculatorSoap')
>)>, {'address': 'http://www.dneonline.com/calculator.asmx'})>)]))>)])
```

In [6]:
```
# podemos ver los métodos disponibles en los servicios:

dir(soap.service)
```

Out[6]:
```
['Add',
 'Divide',
 'Multiply',
 'Subtract',
 '__class__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__get__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__self__',
 '__self_class__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__thisclass__']
```

In [7]:
```
# Vamos a testear el servicio en este caso con la suma de 5 + 5:

soap.service.Add(5, 5)
```

Out[7]:
```
10
```

In [8]:
```
# Vamos a testear el servicio en este caso con la dividir de 4 / 2:

soap.service.Divide(4, 2)
```

Out[8]:
```
2
```

In [9]:
```
# Vamos a testear el servicio en este caso con la multiplicar de  4 * 2:

soap.service.Multiply(4, 2)
```

Out[9]:
```
8
```

In [10]:
```
# Vamos a testear el servicio en este caso con la resta de  4 - 2:
soap.service.Subtract(4, 2)
```

Out[10]:
```
2
```

In [11]:
```
# Nos arroja un error por que el servidor solo espera valores enteros:

soap.service.Subtract(4.2, 2.2)
```

```
---------------------------------------------------------------------------
Fault                                     Traceback (most recent call last)
Cell In[11], line 3
      1 # Nos arroja un error por que el servidor solo espera valores enteros:
----> 3 soap.service.Subtract(4.2, 2.2)

File ~/.local/lib/python3.10/site-packages/zeep/proxy.py:46, in OperationProxy.__call__(self, *args, **kwargs)
     43 if soap_headers:
     44     kwargs["_soapheaders"] = soap_headers
---> 46 return self._proxy._binding.send(
     47     self._proxy._client,
     48     self._proxy._binding_options,
     49     self._op_name,
     50     args,
     51     kwargs,
     52 )

File ~/.local/lib/python3.10/site-packages/zeep/wsdl/bindings/soap.py:135, in SoapBinding.send(self, client, options, operation, args, kwargs)
    132 if client.settings.raw_response:
    133     return response
--> 135 return self.process_reply(client, operation_obj, response)

File ~/.local/lib/python3.10/site-packages/zeep/wsdl/bindings/soap.py:229, in SoapBinding.process_reply(self, client, operation, response)
    227 fault_node = doc.find("soap-env:Body/soap-env:Fault", namespaces=self.nsmap)
    228 if response.status_code != 200 or fault_node is not None:
--> 229     return self.process_error(doc, operation)
    231 result = operation.process_reply(doc)
    233 if message_pack:

File ~/.local/lib/python3.10/site-packages/zeep/wsdl/bindings/soap.py:329, in Soap11Binding.process_error(self, doc, operation)
    326     if child is not None:
    327         return child.text
--> 329 raise Fault(
    330     message=get_text("faultstring"),
    331     code=get_text("faultcode"),
    332     actor=get_text("faultactor"),
    333     detail=fault_node.find("detail", namespaces=fault_node.nsmap),
    334 )

Fault: System.Web.Services.Protocols.SoapException: Server was unable to read request. ---> System.InvalidOperationException: There is an error in XML document (2, 165). ---> System.FormatException: Input string was not in a correct format.
   at System.Number.StringToNumber(String str, NumberStyles options, NumberBuffer& number, NumberFormatInfo info, Boolean parseDecimal)
   at System.Number.ParseInt32(String s, NumberStyles style, NumberFormatInfo info)
   at System.Xml.XmlConvert.ToInt32(String s)
   at Microsoft.Xml.Serialization.GeneratedAssembly.XmlSerializationReader1.Read3_Subtract()
   at Microsoft.Xml.Serialization.GeneratedAssembly.ArrayOfObjectSerializer4.Deserialize(XmlSerializationReader reader)
```

```
   at System.Xml.Serialization.XmlSerializer.Deserialize(XmlReader xmlRead
er, String encodingStyle, XmlDeserializationEvents events)
   --- End of inner exception stack trace ---
   at System.Xml.Serialization.XmlSerializer.Deserialize(XmlReader xmlRead
er, String encodingStyle, XmlDeserializationEvents events)
   at System.Xml.Serialization.XmlSerializer.Deserialize(XmlReader xmlRead
er, String encodingStyle)
   at System.Web.Services.Protocols.SoapServerProtocol.ReadParameters()
   --- End of inner exception stack trace ---
   at System.Web.Services.Protocols.SoapServerProtocol.ReadParameters()
   at System.Web.Services.Protocols.WebServiceHandler.CoreProcessRequest()
```

Creado por:

Isabel Maniega