Informe Taller SOAP Servicios Web

Se lo puede encontrar en GitHub

Integrantes

- Damián Briones
- Nicolás Ontaneda
- Martín Córdova

Instrucciones

Taller Grupal. En los grupos conformados realizar la siguiente actividad.

Cree un servicio Web básico que permita ejemplificar el concepto de SOAP Web Services.

El Servicio lo puede crear en la tecnología de su preferencia (.Net, Java, etc).

Como referencia puede utilizar el video visto en clase sobre ¿Qué es un Servicio Web SOAP?

El estudiante debe subir como evidencia un informe con el taller realizado y el enlace al repositorio Git donde se encuentra el proyecto.

Desarrollo

1. **Elegir el lenguaje de programación o framework a utilizar**

Por facilidad se utilizará python

2. **Instalar Librerias**

Para hacer un servicio SOAP en python se deben instalar las librerias `lxml` y `spyne`

```
pip install lxml
pip install spyne
```

3. **Programación**

Una vez instaladas las librerias, se genera un nuevo archivo python en el cual se realiza la programación. Primero importamos las liberías:

```
from spyne import Application, rpc, ServiceBase, Iterable, Integer, Unicode, Array
from spyne.protocol.soap import Soap11
from spyne.server.wsgi import WsgiApplication
from wsgiref.simple_server import make_server
```

Luego se genera una clase de python, que herede de la clase `ServiceBase` de spyne, esta genera los métodos que pueden ser llamados. En este caso se implementan tres métodos: `say_hello`, `add_numbers` y `fibonacci`.

```
class SoapService(ServiceBase):
    @rpc(Unicode, Integer, _returns=Unicode)
    def say_hello(ctx, name, times):
        return f"Hello, {name}! " * times

    @rpc(Integer, Integer, _returns=Integer)
    def add_numbers(ctx, num1, num2):
        return num1 + num2

    @rpc(Integer, _returns=Unicode)
    def fibonacci_sequence(ctx, n):
        def generate_fibonacci(n):
            fib_sequence = []
            a, b = 0, 1
            for _ in range(n):
                fib_sequence.append(str(a))
                a, b = b, a + b
            return ' '.join(fib_sequence)
        return generate_fibonacci(n)
```

> Nota: el atributo `@rpc` describe los tipos de entrada y el tipo de salida.

Despues podemos generar una nueva aplicacion de `splyne` con las descripcion del servicio

```
application = Application([SoapService],
'your.namespace.here',in_protocol=Soap11(validator='lxml'),out_protocol=Soap11())
```

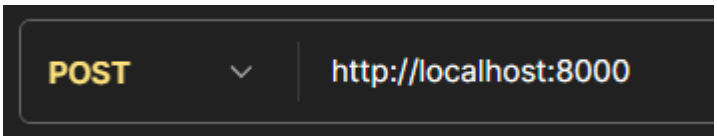
Finalmente, generamos el seevicio en la función `main`

```
if __name__ == '__main__':
    wsgi_application = WsgiApplication(application)

    server = make_server('0.0.0.0', 8000, wsgi_application)
    print("Servicio SOAP iniciado en http://0.0.0.0:8000")
    server.serve_forever()
```

4. **Comprobar la funcionalidad del servicio**

Para llamar al servicio podemos usar `Postman`. Generamos una nueva consulta que llame a `http://localhost:8000`, es importante que sea de tipo `POST`.



Podemos agregar en la pestaña de body, la llamada al servicio seleccionando la pestaña `body` y poniendo el formato en `raw` y seleccionado el tipo `XML`.

- Método: `say_hello`

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:web="your.namespace.here">
   <soapenv:Header/>
   <soapenv:Body>
      <web:say_hello>
         <web:name>Damian</web:name>
         <web:times>3</web:times>
      </web:say_hello>
   </soapenv:Body>
</soapenv:Envelope>
```

- Método: `add_numbers`

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:web="your.namespace.here">
   <soapenv:Header/>
   <soapenv:Body>
      <web:add_numbers>
         <web:num1>5</web:num1>
         <web:num2>7</web:num2>
      </web:add_numbers>
   </soapenv:Body>
</soapenv:Envelope>
```

- Método: `fibonacci`

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:web="your.namespace.here">
   <soapenv:Header/>
   <soapenv:Body>
      <web:fibonacci_sequence>
         <web:n>8</web:n>
      </web:fibonacci_sequence>
   </soapenv:Body>
</soapenv:Envelope>
```

Ejecutamos la consulta en `Postman` y validamos los resultados.

- Método: `say_hello`

```
<?xml version='1.0' encoding='UTF-8'?>
<soap11env:Envelope xmlns:soap11env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="your.namespace.here">
    <soap11env:Body>
        <tns:say_helloResponse>
            <tns:say_helloResult>Hello, Damian! Hello, Damian! Hello, Damian!
</tns:say_helloResult>
        </tns:say_helloResponse>
    </soap11env:Body>
</soap11env:Envelope>
```

- Método: `add_numbers`

```
<?xml version='1.0' encoding='UTF-8'?>
<soap11env:Envelope xmlns:soap11env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="your.namespace.here">
    <soap11env:Body>
        <tns:add_numbersResponse>
            <tns:add_numbersResult>12</tns:add_numbersResult>
        </tns:add_numbersResponse>
    </soap11env:Body>
</soap11env:Envelope>
```

- Método: `fibonacci`

```
<?xml version='1.0' encoding='UTF-8'?>
<soap11env:Envelope xmlns:soap11env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="your.namespace.here">
    <soap11env:Body>
        <tns:fibonacci_sequenceResponse>
            <tns:fibonacci_sequenceResult>0 1 1 2 3 5 8
13</tns:fibonacci_sequenceResult>
        </tns:fibonacci_sequenceResponse>
    </soap11env:Body>
</soap11env:Envelope>
```

Conclusiones

- La implementación de servicios web SOAP proporciona un alto nivel de interoperabilidad entre diferentes plataformas y tecnologías. Al adoptar este estándar, se logra una comunicación efectiva entre sistemas heterogéneos, permitiendo la integración sin problemas de aplicaciones desarrolladas en diversos entornos.
- SOAP proporciona una estructura formal para la comunicación entre servicios, utilizando XML como formato de intercambio de mensajes. Esta formalidad facilita la comprensión y el desarrollo de servicios

coherentes, garantizando una comunicación clara y predecible entre las partes involucradas.
- SOAP ofrece características de seguridad integradas, como WS-Security, que permiten establecer un nivel robusto de protección para las transmisiones de datos. Esto es esencial en entornos donde la seguridad y la integridad de la información son prioritarias, como en aplicaciones empresariales y servicios críticos.

Recomendaciones

- Se recomienda implementar estrategias de optimización del tráfico, como la compresión de mensajes SOAP, para reducir la sobrecarga en las comunicaciones. Esto es especialmente importante en entornos donde los recursos de red son limitados, mejorando la eficiencia de la transferencia de datos.
- Es crucial proporcionar documentación exhaustiva sobre los servicios web SOAP implementados. Esto incluye descripciones detalladas de los métodos disponibles, los parámetros esperados y las respuestas generadas. Una documentación clara facilita la integración de los servicios por parte de otros desarrolladores y equipos.
- Se recomienda establecer un sistema de monitoreo continuo y realizar mantenimientos regulares en los servicios web SOAP. Esto garantizará la detección temprana de posibles problemas, permitiendo correcciones proactivas y asegurando un rendimiento óptimo a lo largo del tiempo. Además, es fundamental mantenerse actualizado sobre las mejores prácticas y posibles actualizaciones en las tecnologías relacionadas con SOAP.