

Desenvolvimento Web II

Aula 05 - Web Service e API

Prof. Fabricio Bizotto

Instituto Federal Catarinense

fabricio.bizotto@ifc.edu.br

Ciência da Computação

13 de janeiro de 2024

1 Web Service

- Definição
- SOAP

Definição

De acordo com a W3C Working Group 2004 diz que é um sistema de software responsável por proporcionar a **interação entre duas máquinas** através de uma rede.

Características

- **Interoperabilidade** - Comunicação entre diferentes plataformas.
- **Independência de Linguagem** - Permite a comunicação entre diferentes linguagens de programação.
- **Formato de Mensagem** - Utiliza XML ou JSON.
- **Padrões Abertos** - Utiliza padrões abertos como SOAP e REST.

Web Service

SOAP

Simple Object Access Protocol

Definição

- Protocolo de comunicação baseado em **XML**.
- As mensagens SOAP basicamente são **documentos XML** serializados seguindo o padrão W3C enviados em cima de um protocolo de rede como HTTP.
- Utiliza **WSDL**, um documento XML que descreve o serviço, especificando como acessá-lo, quais operações executar, quais parâmetros usar, e qual o formato das mensagens.

Estrutura

- *Envelope* - Define o início e o fim da mensagem. É o elemento raiz.
- *Header* - Define informações adicionais sobre a mensagem. Opcional
- *Body* - Define o conteúdo da mensagem. Obrigatório.
- *Fault* - Define informações sobre erros. Opcional

SOAP - Estrutura

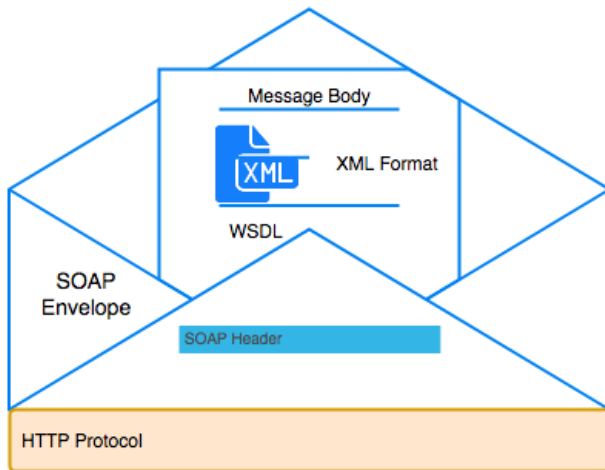


Figura: Estrutura SOAP

Web Service

SOAP - Exemplo

Olá Mundo em SOAP com Python

SOAP - Servidor

```
exemplos > soap > server.py > HelloWorldService > say_hello
1  from spyne import Application, rpc, ServiceBase, Unicode, Integer
2  from spyne.protocol.soap import Soap11
3  from spyne.server.wsgi import WsgiApplication
4  from wsgiref.simple_server import make_server
5
6  class HelloWorldService(ServiceBase):
7
8      # O decorator @rpc define que o método say_hello é um método remoto
9      @rpc(Unicode, Integer, _returns=Unicode)
10     def say_hello(ctx, name, times):
11         ip_address = ctx.transport.req["REMOTE_ADDR"]
12
13         for i in range(times):
14             print(f"Hello {name} from {ip_address} #{i+1}")
15
16     soap_app = Application([HelloWorldService], 'spyne.examples.hello.soap',
17                             in_protocol=Soap11(validator='lxml'),
18                             out_protocol=Soap11())
19
20     # O objeto WsgiApplication é o que o Spyne usa para gerar o servidor WSGI
21     wsgi_app = WsgiApplication(soap_app)
22
23     if __name__ == '__main__':
24         server = make_server('0.0.0.0', 8000, wsgi_app)
25         server.serve_forever()
```

Figura: SOAP - Servidor

SOAP - Cliente

exemplos > soap >  client_soap.py > ...

```
1  from zeep import Client
2  from zeep.plugins import HistoryPlugin
3  from lxml import etree
4
5  # Criar um cliente Zeep com base no URL do WSDL
6  history = HistoryPlugin()
7  client = Client(f'http://localhost:8000/?wsdl', plugins=[history])
8
9  # Chamar o método do serviço
10 response = client.service.say_hello(name='Professor', times=3)
11
12 # Exibir a resposta
13 for hist in [history.last_sent, history.last_received]:
14     print(etree.tostring(hist["envelope"], encoding="unicode", pretty_print=True))
```

Figura: SOAP - Cliente

SOAP - Chamada e WSDL

```
(.venv) fabricio@DESKTOP-MG3SLC3:~/Projetos/Desenvolvimento-Web-II/exemplos/soap$ python server.py
127.0.0.1 - - [12/Jan/2024 12:39:01] "GET /?wsdl HTTP/1.1" 200 2613
Hello Professor from 127.0.0.1
Hello Professor from 127.0.0.1
Hello Professor from 127.0.0.1
127.0.0.1 - - [12/Jan/2024 12:39:01] "POST / HTTP/1.1" 200 235
[]
(.venv) fabricio@DESKTOP-MG3SLC3:~/Projetos/Desenvolvimento-Web-II/exemplos/soap$ python client.py
None
```

```
<wsdl:definitions
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:plink="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
  xmlns:wsdlsoap11="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdlsoap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap11enc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:soap11env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:soap12env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:soap12enc="http://www.w3.org/2003/05/soap-encoding"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
  xmlns:xop="http://www.w3.org/2004/08/xop/include"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:tns="spyne.examples.hello.soap" targetNamespace="spyne.examples.hello.soap" name="Application">
  <wsdl:types>
    <xs:schema targetNamespace="spyne.examples.hello.soap" elementFormDefault="qualified">
      <xs:complexType name="say_hello">
        <xs:sequence>
          <xs:element name="name" type="xs:string" minOccurs="0" nillable="true"/>
          <xs:element name="times" type="xs:integer" minOccurs="0" nillable="true"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="say_helloResponse">
        <xs:sequence>
          <xs:element name="say_helloResult" type="xs:string" minOccurs="0" nillable="true"/>
        </xs:sequence>
      </xs:complexType>
      <xs:element name="say_hello" type="tns:say_hello"/>
      <xs:element name="say_helloResponse" type="tns:say_helloResponse"/>
    </xs:schema>
  </wsdl:types>
  <wsdl:message name="say_hello">
    <wsdl:part name="say_hello" element="tns:say_hello"/>
  </wsdl:message>
```



localhost:8000/?wsdl

Web Service

SOAP - Exemplo com Chamada Direta

Podemos enviar o arquivo XML diretamente para o servidor

SOAP - Código para Chamada Direta com XML

```
exemplos > soap > client_soap_xml.py > ...  
1  from zeep import Client  
2  from zeep.plugins import HistoryPlugin  
3  from lxml import etree  
4  import http.client  
5  
6  # ler o arquivo xml com a requisição  
7  with open("request.xml", "r") as f:  
8      |    xml_content = f.read()  
9  
10 # Criar um cliente Zeep com base no XML  
11 connection = http.client.HTTPConnection("localhost", 8000)  
12 connection.request("POST", "/", xml_content, headers={"Content-Type": "text/xml"})  
13  
14 # Exibir a resposta  
15 response = connection.getresponse()  
16 print(response.status, response.reason)  
17 print(response.read().decode())  
18  
19 # Fechar a conexão  
20 connection.close()
```

Figura: SOAP - Chamada Direta - Cliente

SOAP - Enviando XML para o Servidor via Postman

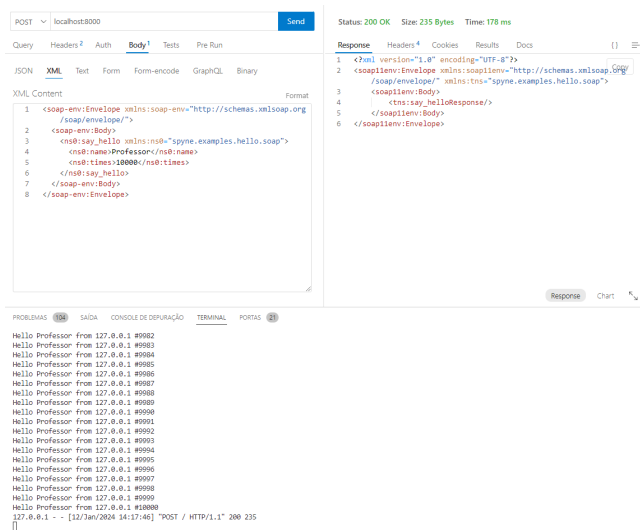


Figura: SOAP - Enviando XML

Web Service

REST

Representational State Transfer

Definição

- É um estilo de arquitetura de software que define a implementação de um serviço web.
- Grande parte dos serviços web usam REST devido a sua simplicidade e desempenho.
- Utiliza os métodos HTTP para definir as operações.
- Diferente de SOAP, não necessita de um documento WSDL ou qualquer outro mecanismo para descrever o serviço.

Fundamentos

- Uma arquitetura baseada em recursos.
- Os recursos são acessados e identificados usando URL.
- Permite que os recursos sejam representados em diferentes formatos, como XML, JSON, HTML, etc.
- **Content negotiation** - O cliente e o servidor negociam o formato de representação do recurso.

REST vs SOAP

SOAP	REST
Geralmente usa HTTP/HTTPS, mas pode usar outros	Usa apenas HTTP/HTTPS
XML	XML, JSON, HTML, etc
Utiliza WSDL	Não utiliza WSDL
Precisa fazer o parse da mensagem	Não precisa fazer o parse da mensagem

Tabela: SOAP vs REST