

Desenvolvimento Web II

Aula 04 - Web Service e API

Prof. Fabricio Bizotto

Instituto Federal Catarinense

fabricio.bizotto@ifc.edu.br

Ciência da Computação
18 de março de 2024

1 Web Service

2 SOAP

■ Experimentos

Definição

- É uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes.
- Permite que aplicações se comuniquem independentemente de linguagem, software e hardware utilizados.
- É uma tecnologia utilizada para **padronizar** e **organizar** a comunicação entre aplicações.

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.

OI, EU ESTOU COM PROBLEMAS AQUI PARA FAZER UMA INTEGRAÇÃO COM O SISTEMA DE VOCÊS E ABRI UM CHAMADO AÍ HÁ ALGUNS DIAS. VOCÊ PODE VERIFICAR?

OK, ESPERA UM POUCO...



É ESSE CHAMADO AQUI SOBRE UM WEBSERVICE?

ESSE MESMO!



ENTÃO, FALEI COM MEU CHEFE AQUI DO T.I. E ELE NÃO CONHECE ESSE SERVIÇO CHAMADO WEBSERVICE... VOCÊ TEM MAIS DETALHES DE QUAIS SISTEMAS ESTÃO ENVOLVIDOS?

FLOFT!



VIDA DE PROGRAMADOR

real historia;
string sender = "Beatriz";

#VIDASNEGRASIMPORTAM



#2019



VIDA DE PROGRAMADOR

.COM.BR

```
real historia;  
string sender;  
sender = "Herbet Mota";
```



#908

AQUI É O PROGRAMADOR,
ESTOU COM UM PROBLEMA AO
ACESSAR UM DOS WEB SERVICES DE
VOCÊS. O RETORNO ESTÁ VINDO
APENAS COM OS DECIMAIS DO
VALOR, SEM A PARTE INTEIRA.



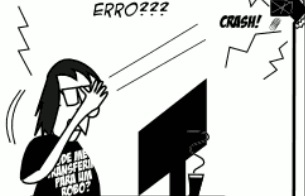
ENTENDO, MAS QUAL FOI A ABA
DO WEB SERVICE QUE O SENHOR
ESTÁ ACESSANDO?

NÃO, VEJA BEM... WEB
SERVICE NÃO TEM ABA.
É UM SISTEMA MEU QUE
ACESSA O SEU SERVIÇO...



HMMM... ENTENDO... MAS QUAL
A PÁGINA DO WEB SERVICE QUE
O SENHOR ESTÁ NAVEGANDO
E ENCONTRANDO O
ERRO???

CRASH!



API - *Application Programming Interface*

API é um termo bastante amplo, que não necessariamente descreve um web service. API é uma interface que permite a comunicação entre dois componentes de software. Por meio de uma API, um software cliente é capaz de consumir serviços disponibilizados por um software servidor.

Web Service

Um web service, por sua vez, é um tipo de API que fornece a sua interface de comunicação **via internet**.

Nem toda API é um web service, mas todos os web services são APIs.

Web Service

SOAP

Simple Object Access Protocol

Definição

- Protocolo de comunicação usado para troca de mensagens entre aplicações.
- As mensagens SOAP basicamente são **documentos XML** serializados seguindo o padrão W3C enviados em cima de um protocolo de rede como HTTP.
- Para descrever os serviços SOAP, é comum utilizar o WSDL (*Web Services Description Language*), um documento XML que define a interface, operações, e protocolos de comunicação.

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

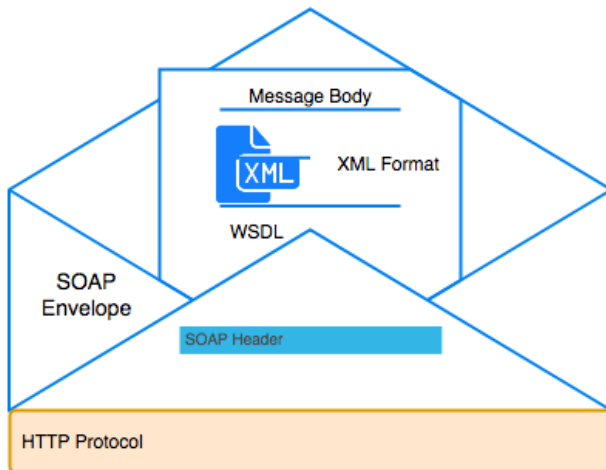


Figura: Estrutura SOAP

Web Service

SOAP - Exemplo

Requisição e Resposta

```

1  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
2    <soapenv:Header/>
3    <soapenv:Body>
4      <sch:UserDetailsRequest>
5        <sch:name>John</sch:name>
6      </sch:UserDetailsRequest>
7    </soapenv:Body>
8  </soapenv:Envelope>

```

```

1  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
2    <soapenv:Header/>
3    <soapenv:Body>
4      <ns2:UserDetailsResponse xmlns:ns2="http://schemas.xmlsoap.org/soap/envelope/">
5        <ns2:User>
6          <ns2:name>John</ns2:name>
7          <ns2:age>5</ns2:age>
8          <ns2:address>Greenville</ns2:address>
9        </ns2:User>
10     </ns2:UserDetailsResponse>
11   </soapenv:Body>
12 </soapenv:Envelope>

```

Figura: SOAP - Exemplo - Requisição e Resposta

Web Service

SOAP - Exemplo

Olá Mundo usando protocolo SOAP e Python

```
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        return f"Hello {name} from {ip_address}!"
17
18 # Criando uma aplicação Spyne com o serviço HelloWorldService
19 soap_app = Application([HelloWorldService], 'spyne.examples.hello.soap',
20                          in_protocol=Soap11(validator='lxml'),
21                          out_protocol=Soap12())
22
23 # Criando um aplicativo WSGI a partir da aplicação SOAP
24 # WSGI: Web Server Gateway Interface é uma especificação padrão para a
25 # interface entre servidores web e aplicações web em Python
26 wsgi_app = WsgiApplication(soap_app)
27
28 if __name__ == '__main__':
29     server = make_server('0.0.0.0', 8000, wsgi_app)
30     server.serve_forever()
```

Figura: SOAP - Servidor

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

```
(.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

Figura: SOAP - Chamada e WSDL


Web Service

SOAP - Exemplo com Chamada Direta

Podemos enviar o arquivo XML diretamente para o servidor

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()
```



```
<soap-env:Envelope
  xmlns:soap-env="http://schemas.xmlsoap.org/soap/en
<soap-env:Body>
  <ns0:say_hello xmlns:ns0="spyne.examples.hello.s
    <ns0:name>Professor</ns0:name>
    <ns0:times>10000</ns0:times>
  </ns0:say_hello>
</soap-env:Body>
</soap-env:Envelope>
```

Figura: SOAP - Chamada Direta - Cliente

SOAP

Enviando XML para o Servidor via Postman

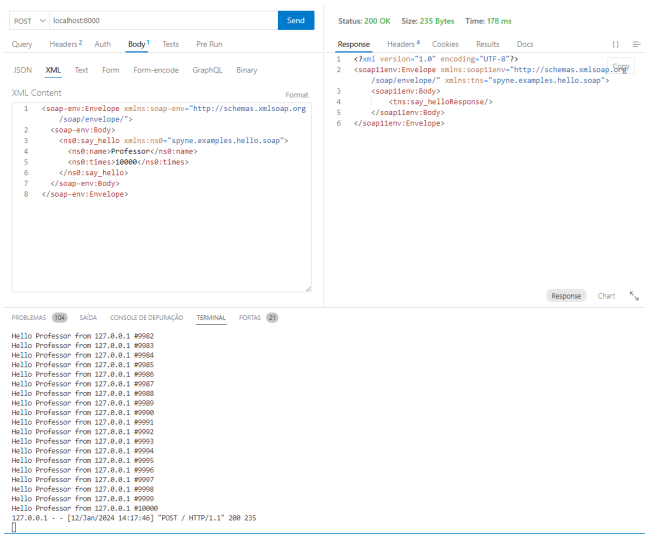


Figura: SOAP - Enviando XML

Experimento 1

- Crie um cliente SOAP para o Web Service <http://www.dneonline.com/calculator.asmx?WSDL>. Escolha uma linguagem de programação de sua preferência.

Experimento 2

- Crie um Web Service do tipo SOAP para calcular o MDC (Máximo Divisor Comum) de uma imagem digital com largura e altura (x e y).
- Para implementar o servidor, use o WSDL disponível aqui.
- Implemente um cliente para testar o Web Service. Com a resposta do servidor, calcule o Aspect Ratio da imagem usando a fórmula:
$$\text{Aspect Ratio} = x/MDC : y/MDC$$
- Peça para outro colega testar seu Web Service.