# Desenvolvimento Web II

## Aula 05 - Web Service e API

Prof. Fabricio Bizotto

Instituto Federal Catarinense

*fabricio.bizotto@ifc.edu.br*

Ciência da Computação

17 de janeiro de 2024

# Roteiro

# Web Service

## 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*

# SOAP

## 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

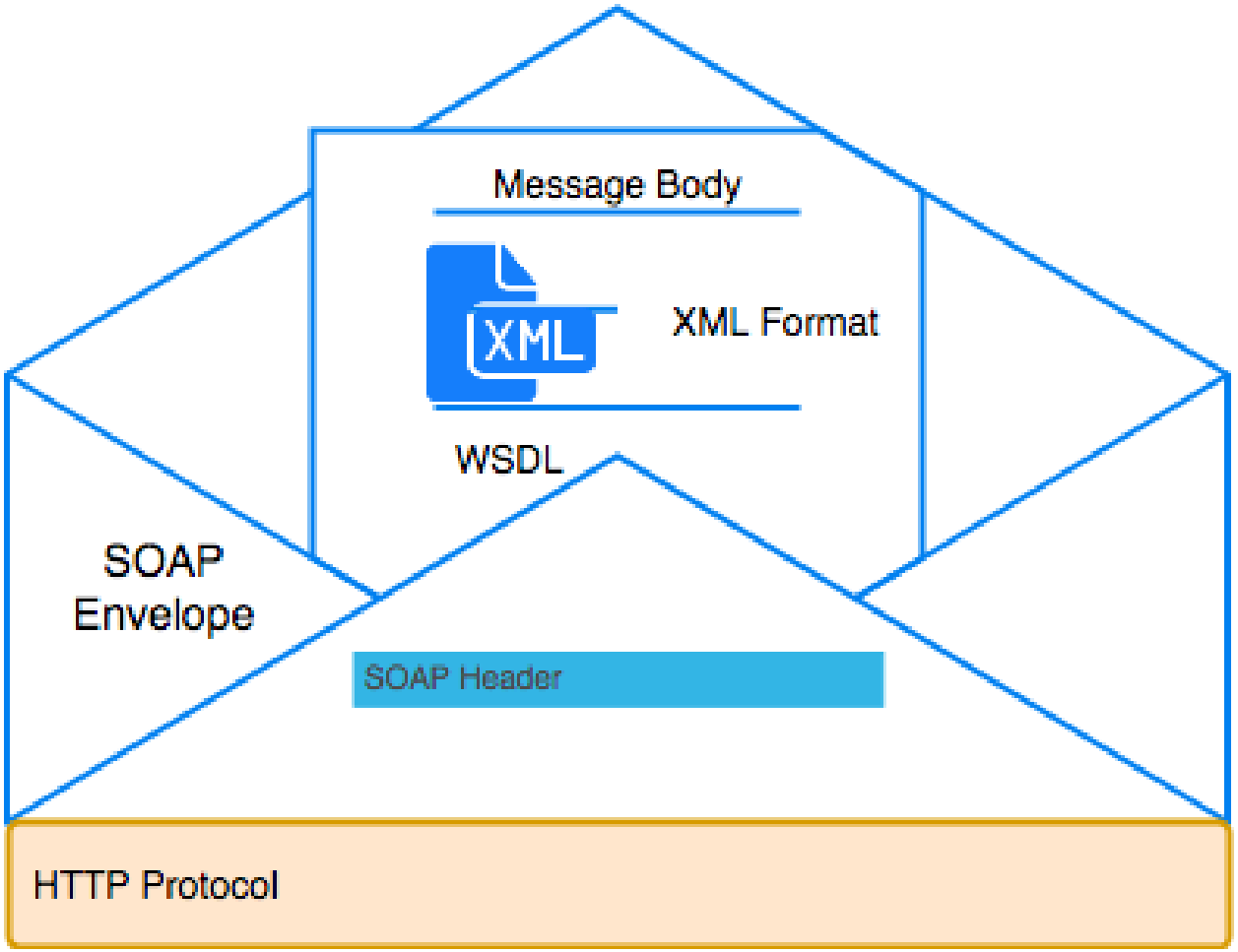


Figura: Estrutura SOAP

Web Service

# SOAP - Exemplo

Olá Mundo em SOAP com Python

# SOAP - Servidor

exemplos > soap > server.py > HelloWorldService > say_hello

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

class HelloWorldService(ServiceBase):

    # O decorator @rpc define que o método say_hello é um método remoto
    @rpc(Unicode, Integer, _returns=Unicode)
    def say_hello(ctx, name, times):
        ip_address = ctx.transport.req["REMOTE_ADDR"]

        for i in range(times):
            print(f"Hello {name} from {ip_address} #{i+1}")

soap_app = Application([HelloWorldService], 'spyne.examples.hello.soap',
                       in_protocol=Soap11(validator='lxml'),
                       out_protocol=Soap11())

# O objeto WsgiApplication é o que o Spyne usa para gerar o servidor WSGI
wsgi_app = WsgiApplication(soap_app)

if __name__ == '__main__':
    server = make_server('0.0.0.0', 8000, wsgi_app)
    server.serve_forever()
```

Figura: SOAP - Servidor

# SOAP - Cliente

exemplos > soap > client_soap.py > ...

```python
from zeep import Client
from zeep.plugins import HistoryPlugin
from lxml import etree

# Criar um cliente Zeep com base no URL do WSDL
history = HistoryPlugin()
client = Client(f'http://localhost:8000/?wsdl', plugins=[history])

# Chamar o método do serviço
response = client.service.say_hello(name='Professor', times=3)

# Exibir a resposta
for hist in [history.last_sent, history.last_received]:
    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
```

localhost:8000/?wsdl

```
<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>
```

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 > ...

```python
from zeep import Client
from zeep.plugins import HistoryPlugin
from lxml import etree
import http.client

# ler o arquivo xml com a requisição
with open("request.xml", "r") as f:
    xml_content = f.read()

# Criar um cliente Zeep com base no XML
connection = http.client.HTTPConnection("localhost", 8000)
connection.request("POST", "/", xml_content, headers={"Content-Type": "text/xml"})

# Exibir a resposta
response = connection.getresponse()
print(response.status, response.reason)
print(response.read().decode())

# Fechar a conexão
connection.close()
```

Figura: SOAP - Chamada Direta - Cliente

# SOAP - Enviando XML para o Servidor via Postman

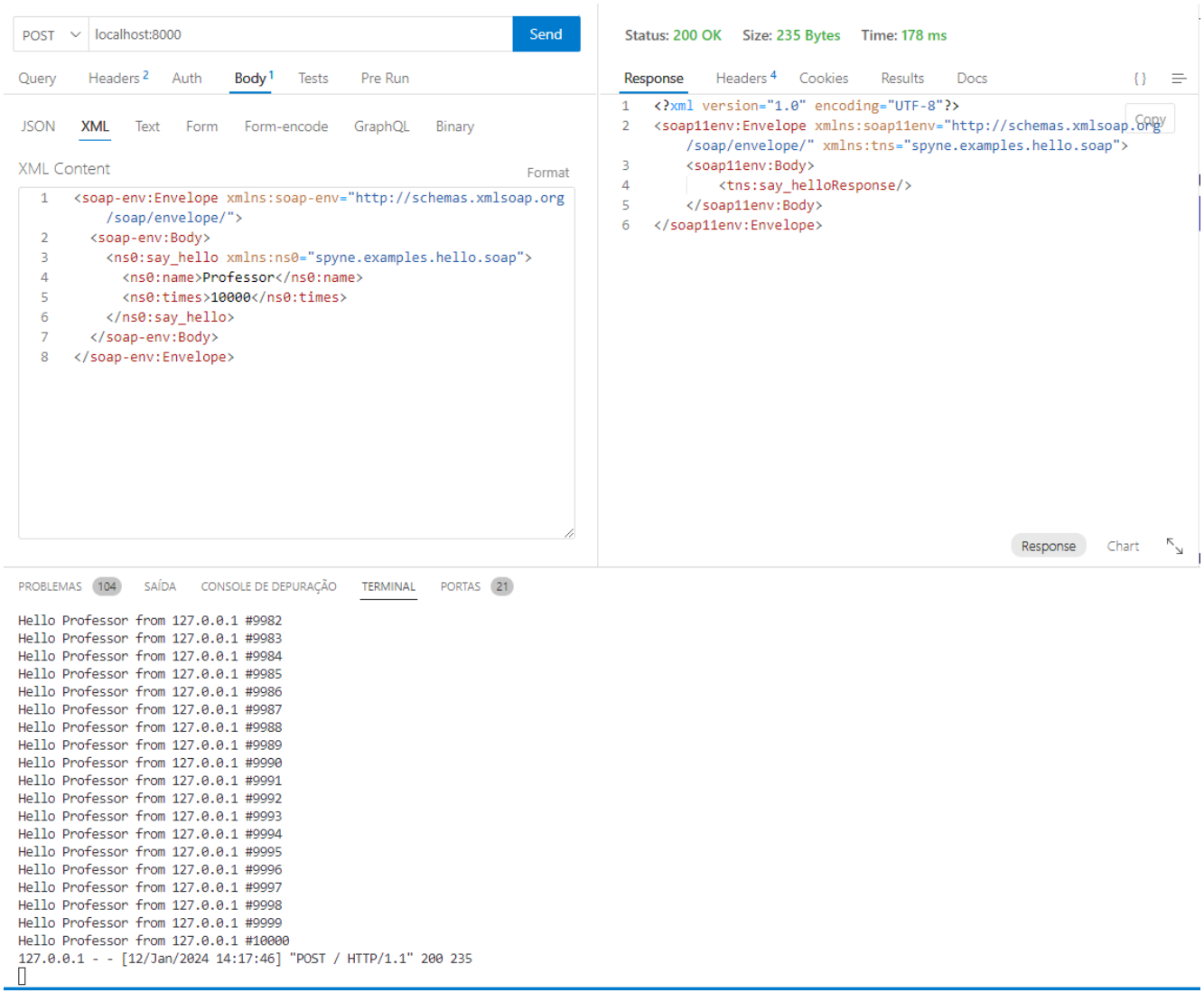


Figura: SOAP - Enviando XML

# Web Service

# REST

*Representational State Transfer*

# REST

## Como surgiu?

A arquitetura de sistema REST foi criada pelo cientista da computação Roy Fielding em 2000.


Anteriormente ele já havia trabalhado na criação do protocolo HTTP e do URI, um conjunto de elementos que identifica recursos nas aplicações web.

Buscando padronizar e organizar os protocolos de comunicação e desenvolvimento na internet, Fielding se uniu a um time de especialistas para desenvolver, durante 6 anos, as características da REST, que foi definida em sua tese de PhD.

# RESTful

## Qual a diferença entre REST e RESTful?

### REST

É uma arquitetura que define um conjunto de princípios para projetar aplicações web. Os critérios que devem ser cumpridos são:

- **Cliente-Servidor** - Separação entre o cliente e o servidor.
- **Stateless** - O servidor não armazena informações sobre o cliente. Cada requisição é independente.
- **Cache** - O servidor deve informar se a resposta pode ser armazenada em cache.
- **Interface Uniforme** - O cliente só precisa saber a URL do recurso e o servidor deve retornar os dados no formato apropriado.
- **Sistema em camadas** - O cliente não precisa saber se está se comunicando diretamente com o servidor ou com um intermediário.

### RESTful

É uma API que implementa os princípios REST.

# 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