Resumen seminario 4

martes, 1 de abril de 2025 19:06

SERVICIOS WEB Y REST API EN PYTHON

Índice

1. Componentes de un sistema distribuido

Un sistema distribuido se compone de diferentes partes o componentes. Cada uno de esos componentes puede estar implementado con un framework o lenguaje de programación diferente. Cada lenguaje está especializado en un tipo de aplicación.

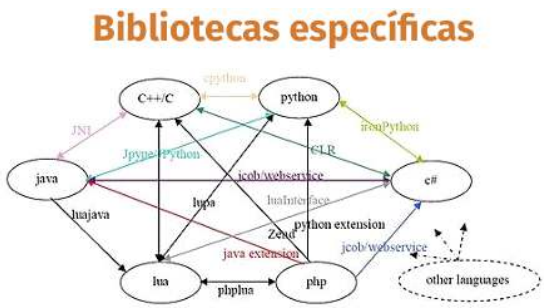
2. Tipos de lenguajes

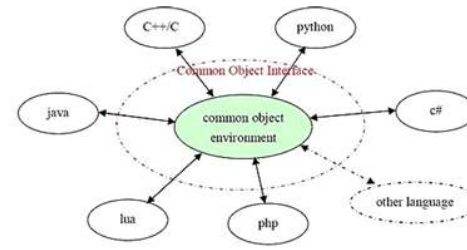
- De bajo, medio o alto nivel
- Multiplataforma (Java o Python)
- Paralelismo y concurrencia
- Plataformas web (PHP o Javascript)
- Tiempo real (Ada)
- Lenguajes “pegamento” (Perl)
- Cálculo científico (Scala)

Usar el mismo lenguaje	Usar distintos lenguajes

• Multiplataforma
• Poseer un extenso conjunto de bibliotecas
	• ¿Comunicación entre componentes?
	  ○ Conexiones
	  ○ Sistema de llamadas remotas
	  ○ Bases de datos

3. Programación multilenguaje





4. Servicios Web (WS) SOAP

Sistema que permite la comunicación y el intercambio de datos entre aplicaciones y sistemas heterogéneos en entornos distribuidos expuestos en una intranet o a través de Internet.

- Estandarizado por el W3C3
- Ofrece un enfoque que permite interoperar a diferentes aplicaciones, sobre diferentes plataformas y/o frameworks.

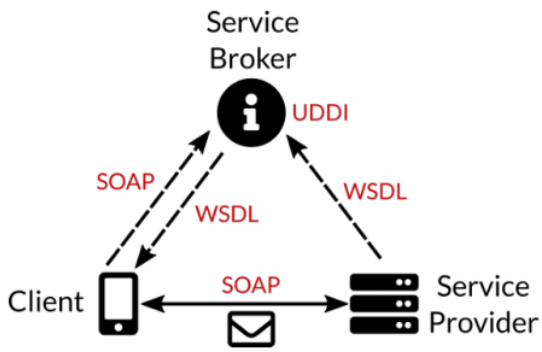
servicios Web SOAP

Exponen la funcionalidad como procedimientos y ejecutables remotos. Las especificaciones están dictadas por los estándares SOAP y WSDL. Tienen el objetivo de solucionar los problemas de integración heredados las tecnologías anteriores (COM, CORBA o RMI) y lograr su interoperabilidad).

Servicios Web REST

Basados en la arquitectura web y en su estándar de base: HTTP. Exponen completamente su funcionalidad como un conjunto coordinado de URIs. Se diseñó para abordar los problemas de SOAP. Permite diferentes formatos de mensajes, como HTML, JSON, XML, y texto plano.

Interacción



5. Estándares de WS SOAP

UDDI (Universal Description Discovery and Integration)

Registro público diseñado para almacenar de forma estructurada información sobre Servicios Web y facilitar su descubrimiento.

WSDL (Web Services Description Language)
Protocolo estándar definido por el W3C para describir un servicio Web (contrato). Describe la interfaz pública de los servicios web:

- Operaciones
- Formatos de mensajes
- Requisitos del protocolo

Lo suelen construir automáticamente las herramientas de desarrollo

SOAP (Simple Object Access Protocol)
Protocolo de comunicación de servicios y aplicaciones web. Establece el formato para enviar y recibir mensajes. Basado en XML e independiente de la plataforma.
Un mensaje SOAP contiene los siguientes elementos:

- **Envelope (obligatorio):** identifica el documento XML como un mensaje SOAP.
- **Header (opcional):** permite extender un mensaje SOAP de forma modular y descentralizada.
- **Body (obligatorio):** contiene la información a transmitir.
- **Fault (opcional):** contiene la información sobre errores y estado

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
   <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
     <n:priority>1</n:priority>
     <n:expires>2001-06-22T14:00:00-05:00</n:expires>
   </n:alertcontrol>
 </env:Header>
 <env:Body>
   <m:alert xmlns:m="http://example.org/alert">
     <m:msg>Pick up Mary at school at 2pm</m:msg>
   </m:alert>
 </env:Body>
</env:Envelope>
```

6. Servicios Web (WS) REST

Interacción

7. Estándares de WS REST

URI (Uniform Resource Identifier)
Identifica un recurso por su nombre, por su ubicación o por ambos. Comprende la URL y/o el URN:

- **URN:** identifica de forma unívoca los recursos electrónicos por un nombre
  - urn:isbn:0451450523
- **URL:** indica un recurso en Internet para poder localizarlo
  - www.uca.es
- **Esquema:** //máquina/directorio/archivo

Comandos HTTP

- **GET:** solicita el recurso ubicado en la URL especificada.
- **HEAD:** solicita el encabezado del recurso ubicado en la URL especificada.
- **POST:** envía datos al programa ubicado en la URL especificada.
- **PUT:** envía datos a la URL especificada.
- **DELETE:** borra el recurso ubicado en la URL especificada

Diferencias entre:
PUT/POST
PUT: pone un recurso en la dirección especificada en la URL y es idempotente.
POST: envía datos a una URL para que el recurso en esa URI los maneje y no es idempotente

JSON(JavaScript Object Notation)

Formato de intercambio de mensajes. Completamente independiente del lenguaje de programación. Consiste en colecciones de pares nombre/valor y listas ordenadas de valores.
Ejemplo de información codificada en JSON | XML

```
{
  "username" : "my_username",
  "password" : "my_password",
  "validation-factors" : {
     "validationFactors" : [
        {
           "name" : "remote_address",
           "value" : "127.0.0.1"
        }
     ]
  }
}
```

```
<authentication-context>
 <username>my_username</username>
 <password>my_password</password>
 <validation-factors>
   <validation-factor>
     <name>remote_address</name>
     <value>127.0.0.1</value>
   </validation-factor>
 </validation-factors>
</authentication-context>
```

8. SOAP VS REST

	SOAP	REST
Diseño	Estandarizado	Pautas y recomendaciones flexibles
Seguridad	Soporte SSL	HTTPS y SSL
Rendimiento	Requiere más recursos	Requiere menos recursos
Mensajes	XML	Texto plano, HTML, XML, JSON, YAML, y otros
Protocolos de Transferencia	HTTP, SMTP, UDP, y otros	HTTP
Recomendado para	Aplicaciones de alta seguridad, servicios financieros, pasarelas de pago	APIs públicas, servicios, móviles, redes sociales
Ventajas	Seguridad y extensibilidad	Flexibilidad, escalabilidad y rendimiento

9. API (Application Programming Interfaces)

Definiciones, subrutinas, funciones, procedimientos y protocolos que ofrece una biblioteca para ser usado por otro software. **Capa de abstracción,** utilizado para **diseñar e integrar** aplicaciones software

10. REST API en Python

Cliente:
Utiliza peticiones HTTP. Sencillo de implementar mediante la biblioteca requests.

Ejemplo de petición GET:

```
import json
import requests
from requests.structures import CaseInsensitiveDict

mykey = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
mycity = "Cádiz"
units = "metric" #Por defecto, la temperatura está en grados Kelvin, para obtener grados Fahrenheit->imperial y Celsius->metric

url = "https://api.openweathermap.org/data/2.5/weather?appid=" + mykey + "&q=" + mycity + "&units=" + units

headers = CaseInsensitiveDict()
headers["Accept"] = "application/json"

resp = requests.get(url, headers=headers)
json_data = json.loads(resp.text)

#Con el código http podemos ver si ha habido algún error
print("Status code: " + str(resp.status_code))

#Mostramos la temperatura en la ciudad indicada
print("Temperature: " + str(json_data["main"]["temp"]) + "ºC\n")

#Mostramos los datos crudos para que se vea todo lo que devuelve el endpoint "weather"
print("Raw data: " + resp.text)
```

https://api.openweathermap.org/data/2.5/**directions/output?parameters**
https://api.openweathermap.org/data/2.5/**service/output?parameters**

Ejemplo petición POST:

```
import requests

API_ENDPOINT = "https://pastebin.com/api/api_post.php"
API_KEY = "XXXXXXXXXXXXXXXX"

source_code = '''
print("Hello, world!")
a = 1
b = 2
print(a + b)
'''

data = {'api_dev_key': API_KEY,
        'api_option': 'paste',
        'api_paste_code': source_code,
        'api_paste_format': 'python'}
r = requests.post(url=API_ENDPOINT, data=data)

pastebin_url = r.text
print("The pastebin URL is:%s" % pastebin_url)
```

Servidor:
Complejo codificarlo a mano mediante sockets. Gestionar peticiones y salida. Gestionar el paralelismo. Múltiples peticiones.
Solución: Framework de desarrollo web

11. Framework Web populares para Python

Pyramid
Flexible, minimalista, rápido y fiable. Fue de los primeros frameworks web compatibles con Python 3. Ideal para desarrollo de aplicaciones web grandes.

Django
Mayor framework web basado en Python. Comunidad grande y activa

Flask
Microframework minimalista de solo un único archivo. Varias extensiones disponibles.

Bottle
Microframework muy simple que proporciona un mínimo de herramientas al desarrollador. Ideal para crear una API web realmente simple.

12. Bottle Framework

Instalar Pip:
Comando para versión 2.x de Python: pip
Comando para versión 3.x de Python: pip3
Instalar pip: sudo apt install python3-pip

Instalar Bottle:
Recomendada: sudo pip3 install bottle

MÁS EJEMPLOS DIAPOSITIVA