1 Python程序设计#3作业

班级: 2022211305

学号: 2022211683

姓名: 张晨阳

1.1 作业题目

基于 aiohttp（https://docs.aiohttp.org/en/stable/） 实现一个服务查询客户端，能够访问#2作业提供的服务。数据获取后进行格式转换：

- JSON结果转换为TEXT格式（字段之间使用空格间隔、记录之间使用换行符间隔）
- XML结果转换为TEXT格式（需求同上）。
- CSV格式转换为TEXT格式（需求同上）。

要求客户端可以通过以上3种格式访问数据服务。

1.2 作业内容

程序源代码嵌入下方的code block中。

```
import aiohttp
import asyncio
import xmltodict

# 服务查询客户端类
class QueryClient:
    def __init__(self, base_url):
        self.base_url = base_url

    async def fetch_data(self, endpoint, params):
        async with aiohttp.ClientSession() as session:
            async with session.get(f"{self.base_url}/{endpoint}", params=params)
as response:
                if response.status == 200:
                    return await response.text()
                else:
                    raise Exception(f"Failed to fetch data: {response.status}")

    def json_to_text(self, json_data):
        import json
        data = json.loads(json_data)
        return "\n".join(f"{entry['Year']} {entry['No_Smoothing']}
{entry['Lowess']}" for entry in data)

    def xml_to_text(self, xml_data):
        parsed_data = xmltodict.parse(xml_data)
        entries = parsed_data["data"]["entry"]
```

```
        if not isinstance(entries, list):
            entries = [entries]  # 确保单个数据也能正常处理
        return "\n".join(f"{entry['Year']} {entry['No_Smoothing']}
{entry['Lowess']}" for entry in entries)

    def csv_to_text(self, csv_data):
        lines = csv_data.strip().split("\n")
        # 忽略CSV头部
        return "\n".join(" ".join(line.split(",")) for line in lines[1:])

    async def query(self, format, start_year, end_year, order):
        endpoint = f"query/{format}"
        params = {
            "start_year": start_year,
            "end_year": end_year,
            "order": order
        }
        raw_data = await self.fetch_data(endpoint, params)

        # 根据格式转换为TEXT
        if format == "json":
            return self.json_to_text(raw_data)
        elif format == "xml":
            return self.xml_to_text(raw_data)
        elif format == "csv":
            return self.csv_to_text(raw_data)
        else:
            raise ValueError(f"Unsupported format: {format}")


# 测试客户端
async def main():
    # 替换为你服务的实际地址
    client = QueryClient("http://localhost:8000")

    # JSON格式查询并转换
    print("JSON -> TEXT")
    json_text = await client.query("json", 1990, 2000, "asc")
    print(json_text)

    # XML格式查询并转换
    print("\nXML -> TEXT")
    xml_text = await client.query("xml", 2000, 2010, "asc")
    print(xml_text)

    # CSV格式查询并转换
    print("\nCSV -> TEXT")
    csv_text = await client.query("csv", 2010, 2020, "asc")
    print(csv_text)

# 启动客户端
if __name__ == "__main__":
    asyncio.run(main())
```

1.3 代码说明

模块和方法说明

`QueryClient` **类**

1. `__init__` **方法**:

- 初始化客户端实例。
- 参数 `base_url`：服务端的基础 URL（如 `http://localhost:8000`）。

2. `fetch_data` **方法**:

- 功能：向服务端发起 GET 请求，获取原始数据。
- 参数:
  - `endpoint`：服务端 API 的具体路径（如 `query/json`）。
  - `params`：请求的查询参数（包括年份范围和排序方式）。
- 实现:
  - 使用 `aiohttp.ClientSession` 进行异步请求。
  - 检查响应状态码，返回响应内容或抛出异常。

3. `json_to_text` **方法**:

- 功能：将 JSON 格式数据转换为 TEXT 格式。
- 参数:
  - `json_data`：从服务端返回的 JSON 数据字符串。
- 实现:
  - 使用 `json.loads` 解析 JSON 字符串。
  - 将每个数据条目按 `"Year No_Smoothing Lowess"` 的格式拼接，每行一个记录，行与行之间使用换行符。

4. `xml_to_text` **方法**:

- 功能：将 XML 格式数据转换为 TEXT 格式。
- 参数:
  - `xml_data`：从服务端返回的 XML 数据字符串。
- 实现:
  - 使用 `xmltodict.parse` 解析 XML 数据。
  - 提取每个记录的 `Year`、`No_Smoothing` 和 `Lowess`，按相同的 TEXT 格式拼接。
  - 确保单条数据也能正确处理（将其转换为列表）。

5. `csv_to_text` **方法**:

- 功能：将 CSV 格式数据转换为 TEXT 格式。
- 参数:
  - `csv_data`：从服务端返回的 CSV 数据字符串。
- 实现:
  - 按行分割 CSV 数据，忽略头部行（列名）。
  - 将每行的字段由逗号分隔改为空格分隔。

6. `query` **方法**：

- 功能：根据指定格式（JSON、XML、CSV）查询服务端数据，并转换为 TEXT 格式。
- 参数：
  - `format`：数据格式（`json`、`xml` 或 `csv`）。
  - `start_year`：查询起始年份。
  - `end_year`：查询结束年份。
  - `order`：排序方式（`asc` 或 `desc`）。
- 实现：
  - 调用 `fetch_data` 获取原始数据。
  - 根据 `format` 调用对应的转换方法（如 `json_to_text`）。
  - 返回转换后的 TEXT 数据。

`main` **函数**

1. 创建 `QueryClient` 实例，指定服务端的基础 URL（`http://localhost:8000`）。
2. 调用 `query` 方法，分别测试 JSON、XML 和 CSV 格式的查询功能：

- JSON 查询：年份范围 1990-2000，升序排序。
- XML 查询：年份范围 2000-2010，升序排序。
- CSV 查询：年份范围 2010-2020，升序排序。

3. 打印每种格式转换后的 TEXT 数据。

示例输出

```
JSON -> TEXT
1992 0.22 0.33
1993 0.23 0.33
1994 0.31 0.34
1996 0.33 0.4
1999 0.38 0.47
2000 0.39 0.5
1991 0.4 0.33
1995 0.44 0.36
1990 0.45 0.33
1997 0.46 0.42
1998 0.6 0.44

XML -> TEXT
2000 0.39 0.5
2001 0.53 0.52
2004 0.53 0.6
2008 0.54 0.64
2003 0.61 0.58
2002 0.62 0.54
2006 0.63 0.62
2009 0.65 0.64
2007 0.66 0.63
2005 0.67 0.61
2010 0.72 0.64
```

```
CSV -> TEXT
2011 0.61 0.66
2012 0.65 0.69
2013 0.67 0.74
2010 0.72 0.64
2014 0.74 0.78
2018 0.85 0.93
2015 0.89 0.83
2017 0.92 0.91
2019 0.97 0.92
2016 1.01 0.87
2020 1.01 0.91
```