报表导出脚本

1. config.py

配置文件，用于存储项目中使用的全局变量和配置信息：

• API 基础 URL 配置

```python
# API配置
BASE_URL = "http://saas.gpsnow.net/saas-pre/web/api"

# 登录配置
LOGIN_CONFIG = {
    "username": "swd",
    "password": "a123456",
    "timezone_second": 28800
}
```

• 支持的语言列表配置

```python
# 支持的语言列表
LANGUAGE_LIST = [
    'en',    # 英语
    'zh-CN', # 简体中文
    'zh-TW', # 繁体中文
    'fr',    # 法语
    'ar',    # 阿拉伯语
    'vi',    # 越南语
    'es',    # 西班牙语
    'pt',    # 葡萄牙语
    'ru',    # 俄语
    'it',    # 意大利语
    'de',    # 德语
    'id',    # 印度尼西亚语
    'fa',    # 波斯语
    'bn',    # 孟加拉语
    'he',    # 希伯来语
    'nl',    # 荷兰语
    'tr',    # 土耳其语
    'geo',   # 格鲁吉亚语
    'pl',    # 波兰语
    'ro',    # 罗马尼亚语
    'sq',    # 阿尔巴尼亚语
    'lo',    # 老挝语
    'mn'     # 蒙古语
]
```

• 文件保存路径配置

```python
# 文件保存路径配置
import os
DESKTOP_PATH = os.path.join(os.path.expanduser("~"), "Desktop")
TRANSLATION_DIR = os.path.join(DESKTOP_PATH, "语言翻译")
```

• 默认请求头配置

```python
# 请求头配置
DEFAULT_HEADERS = {
    'User-Agent': 'python-requests/2.31.0'
}
```

2. login.py

登录模块，为接下来的接口获取 token：

• **设置请求头和请求参数**

```python
def login(username=LOGIN_CONFIG['username'],
          password=LOGIN_CONFIG['password'],
          timezone_second=LOGIN_CONFIG['timezone_second'],
          accept_language='cn'):
    """
    调用登录接口获取token

    Args:
        username (str): 用户名，默认为配置文件中的用户名
        password (str): 密码，默认为配置文件中的密码
        timezone_second (int): 时区秒数，默认为配置文件中的时区设置
        accept_language (str): 接受的语言，默认为'cn'

    Returns:
        str: token字符串，登录失败返回None
    """
    url = f'{BASE_URL}/user-service/user/login'

    # 设置请求头信息
    headers = {
        'accept-language': accept_language,             # 接受的语言
        'user-agent': DEFAULT_HEADERS['User-Agent']     # 用户代理
    }

    # 设置请求参数
    data = {
        'name': username,               # 用户名
        'password': password,           # 密码
        'timeZoneSecond': timezone_second,  # 时区秒数
        'lang': accept_language         # 语言设置
    }
```

• 调用登录 API 获取 token

```python
    try:
        # 发送POST请求
        response = requests.post(url, headers=headers, data=data)
        response.raise_for_status()  # 如果响应状态码不是200，将引发异常

        # 解析响应结果
        result = response.json()
        if response.status_code == 200:  # 请求成功
            return result.get('data', {}).get('token')  # 返回token
        else:
            print(f"登录失败: {result.get('msg')}")  # 打印错误信息
            return None

    except requests.exceptions.RequestException as e:
        # 处理请求异常
        print(f"请求发生错误: {str(e)}")
        if hasattr(e, 'response') and e.response is not None:
            print(f"错误响应状态码: {e.response.status_code}")
            print(f"错误详情: {e.response.text}")
        return None
```

3. get_sta_overview_export.py

调用下载报表接口，创建文件夹保存文件：

• **设置请求头和请求参数**

```python
def get_sta_overview_export(accept_language='',url='', params=None, method="post"):
    """
    调用获取统计概览导出接口，支持多语言导出Excel文件

    Args:
        accept_language (str): 接受的语言代码，如'en'、'zh-CN'等
        url (str): API的完整URL地址
        params (dict): 请求参数字典
        method (str): 请求方法，默认为"post"，支持"get"和"post"

    Returns:
        str: 保存的文件路径，如果发生错误则返回None
    """
    # 设置请求头信息
    headers = {
        'Token': login(accept_language=accept_language),  # 认证Token
        'Accept-Language': accept_language,  # 设置接受的语言
        'User-Agent': DEFAULT_HEADERS['User-Agent']  # 设置User-Agent
    }
```

- 判断请求方式并调用接口

```python
try:
    # 根据指定的方法发送请求
    if "post" == method.lower():
        # 尝试使用JSON格式发送POST请求
        response = requests.post(url, headers=headers, json=params)
        try:
            # 如果JSON请求成功且返回JSON响应，则尝试使用form-data格式重新发送
            if response.ok and response.json():
                response = requests.post(url, headers=headers, data=params)
        except Exception as e:
            pass
    elif "get" == method.lower():
        # 发送GET请求
        response = requests.get(url, headers=headers, params=params)

    # 如果请求不成功，打印详细的请求信息用于调试
    if not response.ok:
        print(f"完整URL: {response.url}")
        print(f"请求方法: POST")
        print(f"请求头: {json.dumps(headers, indent=2, ensure_ascii=False)}")
        print(f"请求参数: {json.dumps(params, indent=2, ensure_ascii=False)}")
        print(f"状态码: {response.status_code}")
        print(f"响应内容: {response.text}")
```

- 创建文件夹并写入文件

```python
        # 按语言创建子目录
        lang_dir = os.path.join(TRANSLATION_DIR, f"导出端-{accept_language}")

        # 创建保存文件的目录（如果不存在）
        os.makedirs(lang_dir, exist_ok=True)

        # 从URL中提取文件名并构建完整的文件路径
        file_name = os.path.join(lang_dir, f"{url.split('/')[-1]}_download.xlsx")

        # 将响应内容保存为Excel文件
        with open(file_name, "wb") as f:
            f.write(response.content)

        # 检查响应状态码，如果不是2xx则抛出异常
        response.raise_for_status()

        print(f"文件已成功保存到: {file_name}")
        return file_name

    except requests.exceptions.RequestException as e:
        # 处理请求异常
        print(f"请求发生错误: {str(e)}")
        if hasattr(e, 'response') and e.response is not None:
            print(f"错误响应状态码: {e.response.status_code}")
            print(f"错误响应头: {json.dumps(dict(e.response.headers), indent=2, ensure_ascii=False)}")
            print(f"错误详情: {e.response.text}")
        return None
    except IOError as e:
        # 处理文件操作异常
        print(f"文件操作错误: {str(e)}")
        return None
```

• 接口列表和参数列表

```
    # 定义API端点和参数列表
    url_list = [
        # 统计报表-运行统计
        # 运行总览
        (f"{BASE_URL}/location-service/position/getStaOverviewExport",
         {"endTime": "2025-06-07 16:00:00", "entId": "1925747610618953728", "startTime": "2025-06-06 16:00:00"}),
         #里程统计1
         (f"{BASE_URL}/location-service/position/mileageStaByDayExport",
         {"carId":"1881317797200396288", "endTime": "2025-06-07 16:00:00", "entId": "1925747610618953728", "startTime": "2025-06-06 16:00:00"}),
         #超速详单2
         (f"{BASE_URL}/location-service/position/getOverSpeedDetailExport",
         {"carId":"1881317797200396288", "endTime": "2025-06-07 16:00:00", "entId": "1925747610618953728", "startTime": "2025-06-06 16:00:00"}),
         #停留详单3
         (f"{BASE_URL}/location-service/position/getStopDetailExport",
         {"carId":"1881317797200396288", "mapType":"1", "endTime": "2025-06-07 16:00:00", "entId": "1925747610618953728", "startTime": "2025-06-06
          "GET"),
         #ACC统计4
         (f"{BASE_URL}/device-service/accSta/queryDetailWithUserOrCarIdExport",
         {"carId":"1881317797200396288", "mapType":"1", "endTime": "2025-06-07 16:00:00", "entId": "1925747610618953728", "startTime": "2025-06-06
          "GET"),
         #行程统计5
         (f"{BASE_URL}/location-service/position/distanceStaExport",
         {"carId":"1881317797200396288", "intervalTime":"180", "selectKey":"mileage", "mapType":"1", "endTime": "2025-06-07 16:00:00", "entId": "1
         #离线统计6
         (f"{BASE_URL}/device-service/car/queryOffineCarExport",
         {"entId": "1881159263183699968","duration":"7","prependFilterType":"2","apppendFilterType":"2","recursion":"false","mapType":"1","minInte
         #总速统计7
        (f"{BASE_URL}/location-service/position/getIdlingDetailExport",
         {"carId":"1881317797200396288", "mapType":"1", "endTime": "2025-06-10 15:59:59", "entId": "1881159263183699968", "startTime": "2025-06-05
         #静止统计8
        (f"{BASE_URL}/device-service/structure/getStaticStatisticsExport",
         {"entId":"1881159263183699968","subFlag":False,"flag":5,"timeInterval":-1,"mapType":1,"startTime":"2025-06-08 16:00:00","endTime":"2025-0
```

4. get_task_list.py

报告列表获取模块：

• 设置请求头和请求参数

```
def get_task_list(accept_language='',):
    """获取导出任务列表

    Args:
        accept_language (str): 接受的语言代码，如'zh-CN', 'en'等

    Returns:
        set: 包含(taskId, url)元组的集合，如果发生错误则返回None
    """
    # API端点URL
    url = f'{BASE_URL}/device-service/task/list'

    # 设置分页参数
    params = {
        'pageIndex': 1,  # 页码，从1开始
        'pageSize': 20   # 每页显示的记录数
    }

    # 设置请求头信息
    headers = {
        'accept': 'application/json, text/plain, */*',
        'Accept-Language': accept_language,
        'token':login(accept_language=accept_language),
        'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36'
    }
```

• 发送 GET 请求获取任务列表

```python
    try:
        # 发送GET请求获取任务列表
        response = requests.get(url, params=params, headers=headers)

        # 解析JSON响应
        json_response = response.json()

        # 初始化返回数据集合
        return_data = set()

        # 从响应中提取任务信息
        if 'data' in json_response:
            data = json_response['data']
            if isinstance(data, list):
                # 遍历任务列表，提取taskId和下载URL
                for idx, task in enumerate(data, 1):
                    return_data.add(
                        (task['taskId'], task['url'])
                    )
        return return_data

    except requests.exceptions.RequestException as e:
        # 处理网络请求异常
        print(f"请求发生错误: {e}")
        return None
    except json.JSONDecodeError as e:
        # 处理JSON解析异常
        print(f"JSON解析错误: {e}")
        return None
```

5. download_report.py

创建报表下载任务并下载文件：

• **设置请求头和请求参数**

```python
def download_report(accept_language='', base_url='', params=None):
    """
    调用获取统计概览导出接口，并下载生成的报表文件

    Args:
        accept_language (str): 接受的语言代码，如'zh-CN', 'en'等
        base_url (str): API的基础URL地址
        params (dict): 请求参数，包含导出配置信息
        method (str): 请求方法，'get'或'post'，默认为'get'

    Returns:
        str: 保存的文件路径，如果失败则返回None
    """
    # 获取当前任务列表，用于后续对比找出新建的任务
    old_task_list = get_task_list()

    # 设置请求头信息
    headers = {
        'Token':login(accept_language=accept_language),
        'Accept-Language': accept_language,
        'User-Agent': 'python-requests/2.31.0'
    }
```

• 调用接口创建下载任务

```
        # 下载文件
        response = requests.get(url)

        # 设置保存文件的路径结构
        base_dir = os.path.join(os.path.expanduser("~"), "Desktop", "语言翻译")
        lang_dir = os.path.join(base_dir, f"导出端-{accept_language}")
        os.makedirs(lang_dir, exist_ok=True)

        # 构建文件名并处理特殊字符
        file_name = os.path.join(lang_dir, rf"{url.split('/')[-1]}").replace(" ", "_").replace(":", "_").replace("C_", "C:")

        # 保存文件
        with open(file_name, "wb") as f:
            f.write(response.content)

        # 检查下载是否成功
        response.raise_for_status()

        print(f"文件已成功保存到: {file_name}")
        return file_name

    except requests.exceptions.RequestException as e:
        # 处理请求异常
        print(f"请求发生错误: {str(e)}")
        if hasattr(e, 'response') and e.response is not None:
            print(f"错误响应状态码: {e.response.status_code}")
            print(f"错误响应头: {json.dumps(dict(e.response.headers), indent=2, ensure_ascii=False)}")
            print(f"错误详情: {e.response.text}")
        return None
    except IOError as e:
        # 处理文件操作异常
        print(f"文件操作错误: {str(e)}")
        return None
```