網路程式設計
Web API

Instructor: 馬豪尚

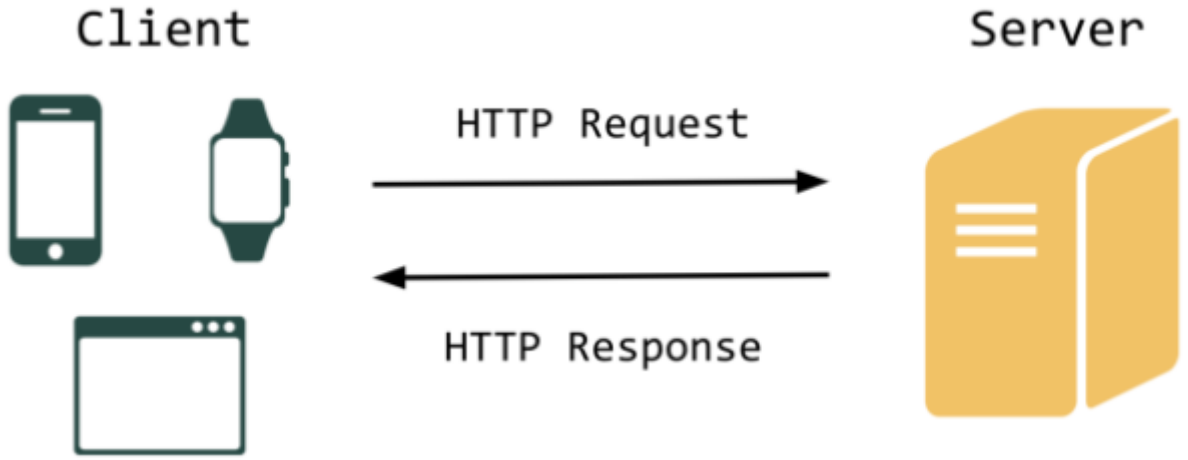
應用程式介面
Application Programming Interface

Application Programming Interface

- 扮演著應用程式和應用程式之間，交換資訊的溝通橋樑
- 定義多個軟體中介之間的互動方式
  - 僅定義了一個介面，而不涉及應用程式在實作過程中的具體操作
  - 可以進行的呼叫（call）或請求（request）的種類
  - 如何進行呼叫或發出請求應
  - 使用的資料格式
  - 應遵循的慣例

Web API

- 在 Web Application 的開發情境下的 API 被稱為 Web API
- 客戶端和伺服器端會透過 HTTP 通訊協定來進行請求與回應



Google Map API

- 建立專案並加入API
  - 進到 Google API程式庫頁面
  - 搜尋關鍵字 "Maps"
  - 找到要使用的API (如有Places API, Geocoding API)，進入之後再按啟用
  - Google會要求你建立API專案，按建立進入後，隨便取個名字新增專案。
  - 重複步驟加入需要的API到專案裏頭，並確認API有正確加入專案

Google Map API

- 取得API的key
  - 進入Google Maps Plateform 開發者頁面來取得API Key

在Python中使用Google Map API

- 安裝Python Client for Google Maps Services
  - pip3 install googlemaps
- 載入模組
  - import googlemaps
- 宣告map物件
  - gmaps=googlemaps.Client(key="你的api key")

Google Map
Geocode API核心功能

Geocode API

- Geocode API主要提供地址、經緯度座標或地點 ID的查詢
  - 協調地理編碼
  - 反向地址查詢
  - 地點 ID 地理編碼
- 返回地址或地點的地理編碼資料
  - 地址的地理座標
  - 經緯度座標組的地址
  - 地點 ID 的地址

Geocode API查找地點

- 協調地理編碼geocode(address)
  - 取得地址的地理座標
  - 參數可以輸入一個地址
  - 參數可以輸入一個地名
- Example
  - geocode_result = gmaps.geocode('1600 Amphitheatre Parkway, Mountain View, CA')
  - geocode_result = gmaps.geocode("臺中市")

Geocode API查找地點

- 反向地址編碼reverse_geocode((x, y))
  - 取得一組經緯度座標的地址
  - (x, y)為一個經緯度座標

- Example
  - reverse_geocode_result = gmaps.reverse_geocode((40.714224, -73.961452))

地理編碼與反向地理編碼結果

- 返回結果為一個json格式，一般結果包含下列欄位
  - address_components[]
    - 這個陣列包含這個地址適用的各種元件
  - formatted_address
    - 這個字串包含這個地點的清楚易懂地址
  - geometry{}
    - 包含經過地理編碼的經緯度值以及地點id等

地理編碼與反向地理編碼結果

- address_components[]
  - 每個地址元件通常會包含下列欄位：
  - types[] 是一個陣列，用來指出地址元件的「類型」。請參閱支援類型清單。
  - long_name 是地理編碼器所傳回地址元件的完整文字說明或名稱。
  - short_name 是地址元件的縮寫文字名稱 (如有)。舉例來說，阿拉斯加州地址元件的 long_name 可能為「Alaska」，而 short_name 則為 2 個字母的郵政簡碼「AK」。

地理編碼與反向地理編碼結果

- types[] 陣列表示「地址類型」
  - 地址類型包括街道地址、國家/地區或政治實體
  - 地址可能有多種類型。這些類型可視為「代碼」。舉例來說，許多城市都會加上 political 和 locality 類型標記。
    - 'types': ['country', 'political']}]
    - 'types': ['administrative_area_level_1', 'political']}

完整的Type地址類型代碼

代碼	說明
street_address	表示精確的街道地址
route	表示具名道路 (例如「國道一號」)
intersection	表示主要的十字路口，通常有兩條主要道路交會
political	表示政治實體。通常表示某些行政管理區的多邊形區域
country	表示國家政治實體，且通常是地理編碼器所傳回的最高順位類型
administrative_area_level_(1-7)	表示國家/地區層級底下的第1-7順位行政實體。在美國境內，1級指州，2級指郡。部分國家沒有這類行政層級
colloquial_area	表示實體的常用替代名稱
locality	表示自治城市或鄉鎮的政治實體
sublocality	表示縣市底下的第一順位行政實體
neighborhood	表示具名社區
premise	表示具名地點，通常是建築物或具有共同名稱的建築物群
subpremise	表示具名地點底下的第一順位實體，通常是具有共同名稱的建築物群中的單一建築物

完整的Type地址類型代碼

代碼	說明
plus_code	表示經過編碼的位置參照，衍生自經緯度
postal_code	表示國家/地區郵政地址所使用的郵遞區號
natural_feature	表示明顯的自然地貌
airport	表示機場
park	表示具名公園
point_of_interest	表示具名搜尋，點例如「帝國大廈」或「艾菲爾鐵塔」
parking	表示停車場
post_box	表示特定郵政信箱
bus_station	表示公車停靠站
train_station	表示火車停靠站
transit_station	表示大眾運輸停靠站
landmark	表示附近地點

地理編碼與反向地理編碼結果

- geometry 包含下列資訊：
  - location 包含經過地理編碼的經緯度值。進行一般地址查詢時，這個欄位通常是最重要的。
  - location_type 會儲存指定位置的其他相關資料，目前支援下列值：
    - "ROOFTOP" 表示傳回結果是精準的地理編碼，以便我們取得準確的街道地址資訊。
    - "RANGE_INTERPOLATED" 表示傳回結果反映了插入在兩個精確點之間 (例如十字路口) 的約略位置 (通常是在道路上)。如果街道地址沒有精準的地理編碼，系統通常就會傳回插入的結果。
    - "GEOMETRIC_CENTER" 表示傳回結果是結果的幾何圖形中心，包括折線 (例如街道) 或多邊形 (區域)。
    - "APPROXIMATE" 表示傳回結果是約略位置。
  - place_id是地點的id值

Geocode API查找地點

- 用地點id來取得資訊gmaps.geocode(place_id=“id”)
  - 輸入為地點的id
- Example
  - gmaps.geocode(place_id='ChIJ7yJ5-d8XaTQRf0SmfuQ-Uoc')

Google Map
Places API核心功能

Places API核心功能

- 地點查尋
  - 搜尋建築物、重要搜尋點的相關資訊
- 地點詳細資訊
  - 搜尋建築物、重要搜尋點的詳細資訊
- 地點相片
  - 在應用程式中新增地點的高畫質相片

Places API核心功能

- 查詢自動完成
  - 為應用程式新增即時地理查詢預測 (使用文字查詢功能，例如「我附近的披薩店」)。
- 地點 ID
  - 取得特定地點的詳細資料，找出特定地點的 ID，並進一步瞭解如何儲存及重新整理地點 ID。
- 地點類型
  - 利用地點類型來限制地點搜尋和自動完成要求的結果。

Places API地點查尋

- find_place('address', 'textquert/phonenumber', fields=[], location_bias)
  - 第一個address的參數可以輸入一個地址, 地名或是電話號碼
  - 第二個參數是代表第一個參數的類型
    - 輸入是地址或地名要設定為'textquert'
    - 輸入是電話號碼要設定為'phonenumber'
  - 第三個參數Fields是一個多個參數的list
    - 代表想要查詢該地點的什麼相關資訊
  - 第四個參數location_bias為查詢範圍的參考值
    - 可以使用一個圓來表示 → circle:radius@lat,lng
      - radius為半徑(公尺), lat,lng為圓中心經緯度
    - 可以使用一個四邊形來表示
      - 用兩個經緯度的座標來表示四邊形的左上和右下(x, y)

Places API地點查尋

› Fields可以指定返回以下的資訊值

類型	資料型態	說明
business_status	String	是否營運中
formatted_address	String	地址
geometry	Geometry	經緯度資訊
icon	String	圖示(回傳圖片資源網址)
name	String	地點名稱
photo	String	照片(回傳google map網址)
place_id	String	地點id
opening_hours	Dict	是否營業中
rating	Number	評分

Places API地點查尋

- Example
  - gmaps=googlemaps.Client(key=“key")
  - gmaps.find_place('洲際棒球場', 'textquery', fields=["business_status","icon","photos","formatted_address", "name","geometry", "place_id"])

Places API 查詢地點詳細資訊

- places(address)
  - address的參數為地址或地名
  - 返回該地點的全部詳細資訊
- Example
  - gmaps=googlemaps.Client(key="key")
  - gmaps.places('804高雄市鼓山區美術館路80號')

Places API 鄰近地區查詢

- places_nearby(location, radius, keyword, min_price, max_price, open_now, rank_by="distance")
  - location={'lat': 緯度值, 'lng': 經度值}
  - radius=搜尋半徑(預設50,000m)
  - Keyword=關鍵字
  - min_price, max_price = 0-4(0表示最實惠-4表示最貴)
  - open_now=True/Flase
  - rank_by="distance"表示由距離排序, 當使用這個模式的時候搜尋半徑參數會失效

Places API 鄰近地區查詢

- Example
  - gmaps=googlemaps.Client(key=“key")
  - gmaps.places_nearby(keyword="餐廳", location=loc, rank_by="distance", min_price=1, max_price=4,)

Places API 地點id查詢

- 地點 ID查詢place(place_id, fields=[], reviews_no_translations, reviews_sort)
  - 用id取得特定地點的詳細資料
  - 可以取得地點的評論
- Example
  - gmaps=googlemaps.Client(key=“key")
  - gmaps.place("ChIJN1t_tDeuEmsRUsoyG83frY4", fields=["business_status", "geometry/location", "place_id", "reviews"], reviews_no_translations=True, reviews_sort="newest")

Places API 地點相片搜尋

- places_photo(ref, max_width=100)
  - ref為相片的參考位置
  - 可以在find_place裡Fields得到
    - 'photo_reference': 'AZose0kIqtp373BbbsYpQEmQNM7jnpam1es2_JNCNs8gUXr4C3hbE3_3eT7kWR22zWWiEfvGUueRg0w6UYA_N3CFy7XDDwHFfcFAb_XOSq4FYHF4q9p9MxNxU8TARFMiiLWg9aDdxzTYIy8F0FNGrM1-kN2RhW51WVRlnbgUl6j6_YTtDi49'
  - 返回結果為二進制的圖片編碼資料

Places API自動完成查詢

- places_autocomplete_query(“查詢關鍵字")
  - 查詢自動完成服務提供使用者不必搜尋特定地點
  - 可以根據類別進行搜尋 (例如「紐約市附近的披薩店」)，並且利用與字串相符的建議查詢來回應
- Example
  - gmaps=googlemaps.Client(key=“key")
  - gmaps.places_autocomplete_query("pizza near New York")

練習

- 運用Google Map API來規劃並推薦一日台北旅遊行程
  - 查詢第一個景點台北101，從這個景點開始你的一日行程規劃
  - 行程規劃
    - 早餐餐廳→推薦景點→午餐餐廳→推薦景點→晚餐餐廳→住宿飯店
  - 每個景點和餐廳之間相互的距離不能超過20公里
  - 推薦原因不限制，可以根據評分、距離等等
- 請將規劃好的行程裡的景點和餐廳資訊儲存起來
  - 至少要包含景點名稱、景點地址、景點照片等資訊