Statistisches Bundesamt

# GENESIS-Online

## Beispiele für POST-Anfragen an die RESTful/JSON-Schnittstelle mit Python

Um Sie bei der Erstellung von POST-Anfragen zu unterstützen, stellen wir lauffähige Code-Beispiele mit der Programmiersprache Python zur Verfügung. Diese Codebestandteile sind ein einmaliger Service.

Erschienen im Mai 2025

# Genesis API: Änderungen zum 30. Juni 2025

Zur Verbesserung des Schutzes Ihrer Nutzerdaten, wird die bisher angebotene Möglichkeit der API Nutzung mittels `GET` Requests am 30. Juni 2025 endgültig abgeschaltet. Nach diesem Termin ist die Genesis-API ausschließlich mit der `POST` Methode erreichbar. Dabei werden die Zugangsdaten im Header der Anfrage gesendet und können damit nicht in Logfiles gelangen oder beim Transport abgegriffen werden.
Um Sie bei der Umstellung zu unterstützen, zeigen wir im Folgenden lauffähige Code-Beispiele, die die POST Methode verwenden, beginnend mit der Programmiersprache **Python** und dem Statistik-Paket **Pandas**.

In [54]:
```
import requests
import pandas as pd
import io
import zipfile
```

Wir empfehlen im Folgenden zur Identifizierung gegenüber dem Webservice den **API Token**. Diese Zeichenkette finden Sie nach dem Login im Menü Webservice-Schnittstelle (API). Er lässt sich unabhängig von Ihren Zugangsdaten zurücksetzen und bietet Ihnen damit mehr Kontrolle bei gemeinsam genutzten Projekten, in Schulungsumgebungen usw. Ebenfalls enthält der Token keine Zeichen, die ggf. maskiert werden müssten und ist somit in der Anwendung weniger fehleranfällig.
Allerdings sind nach einer Identifizierung mit Token keine schreibenden Zugriffe möglich. Dies ist zum einen die Änderung des Passworts per API oder eine Tabellenabfrage, die aufgrund ihrer Größe in eine Warteschlange oder Batch geschrieben wird ( `job=true` ). Solche schreibenden Zugriffe müssen mittels Nutzername/ E-Mail und Passwort identifiziert werden.
Die hier beschriebenen Programmbeispiele können für beide Identifizierungsarten genutzt werden.
Sollten Sie noch keine Zugangsdaten für die Genesis Datenbank haben, können Sie sich einfach und kostenfrei registrieren.

In [66]:
```
BASE_URL = 'https://www-genesis.destatis.de/genesisWS/rest/2020/'

TOKEN = "58b3_IHREN_TOKEN_EINFÜGEN_e6d032"
```

Unabhängig von der Art der verwendeten Zugangsdaten müssen diese im request header einer POST Anfrage mit dem `'Content-Type': 'application/x-www-form-urlencoded'` übermittelt werden.

In [5]:
```
headers = {
    'Content-Type': 'application/x-www-form-urlencoded',
    'username': TOKEN,
    'password': ""
    }

langPref = "de"
```

## Zugangsdaten und API-Verfügbarkeit testen (optional)

Es ist hilfreich, zunächst die Zugangsdaten sowie die Verbindung zur API zu testen. Dies ist zugleich die einfachste Methode, bei der als Parameter – neben der Identifizierung – lediglich die gewünschte Sprache (de|en) übergeben wird.

In [6]:
```
hello = requests.post(BASE_URL + 'helloworld/logincheck',
    headers = headers,
    data = {
    'language': langPref
    })
```

In [69]:
```
hello.text[:-36]
```

Out[69]:
```
'{"Status":"Sie wurden erfolgreich an- und abgemeldet! Bei mehr als 3 parallelen Requests wurden länger als 15 Minuten laufende Requests beendet.","Username"'
```

## Download einer einzelnen Tabelle im Excel-Format

Als Beispiel sollen einzelne COICOP-5-Steller ( `CC13A5` ) des monatlichen Verbraucherpreisindex abgerufen werden, und zwar alle, die zu `CC13-071 Kauf von Fahrzeugen` gehören.
Eine Dokumentation des gesamten Funktionsumfangs der API finden Sie in der sprachliche Schnittstellenbeschreibung (PDF, 899kB)

In [24]:
```
responseTable = requests.post(BASE_URL + 'data/tablefile',
    headers = headers,
    data = {
    'name': '61111-0004',
    'startyear': 2025,
    'transpose': "true",
    'classifyingvariable1': "CC13A5",
    'classifyingkey1': "CC13-071*",
    'compress': 'true',
    'format': 'xlsx',
    'language': langPref
    })
```

### Abspeichern der Datei, sofern der Abruf erfolgreich war

In [27]:
```
if responseTable.status_code == 200:
    with open("table.xlsx", "wb") as f:
        f.write(responseTable.content)
else:
    print("Antwort-Code: "+(responseTable.status_code))
```

## Umgang mit großen Tabellen

Tabellen mit mehr als 40.000 Werten (Stand 24. März 2025) können per API nicht im Dialog direkt heruntergeladen werden. Entweder muss die Anfrage in eine Warteschlange ( `job=true` ) geschickt und später mit `resultfile` abgeholt werden oder man splittet die Anfrage auf.
Das Beispiel der monatlichen COICOP-10-Steller auf Basis von 2020 überschreitet diese Grenze bereits Anfang 2025:

In [47]:
```
response = requests.post(BASE_URL + 'data/tablefile',
    headers = headers,
    data = {
    'name': '61111-0006',
    'startyear': 2020,
    "classifyingvariable1": "CC13Z1",
    'compress': 'true',
    'format': 'ffcsv',
    'language': langPref
})
# prüfe, ob die Antwort Text, binär oder leer ist
print(response.text[0:120])
```

```
{"Ident":{"Service":"data","Method":"tablefile"},"Status":{"Code":98,"Content":"Die Tabelle ist zu gross
, um im Dialog a
```

### Aufteilung in zwei Tabellen entlang der Zeitachse

Für die Weiterverarbeitung in Pandas ist das Flatfile-csv-Format ( `ffcsv` ) das Mittel der Wahl. Diese Dateien können einfach hintereinander zusammengeführt werden. In vielen Fällen entspricht die Zahl der Zeilen der Werteanzahl. Im folgenden Beispiel werden zuerst die Werte für Januar 2020 bis Dezember 2023 abgerufen, danach der aktuelle Rand (Januar 2024 und später).

In [ ]:
```
response1 = requests.post(BASE_URL + 'data/tablefile',
    headers = headers,
    data = {
    'name': '61111-0006',
    'startyear': 2020,
    'endyear': 2023,
    "classifyingvariable1": "CC13Z1",
    'compress': 'true',
    'format': 'ffcsv',
    'language': langPref
})
```

In [ ]:
```
response2 = requests.post(BASE_URL + 'data/tablefile',
    headers = headers,
    data = {
    'name': '61111-0006',
    'startyear': 2024,
    "classifyingvariable1": "CC13Z1",
    'compress': 'true',
    'format': 'ffcsv',
    'language': langPref
})
```

### Gezippte Datei-Antwort direkt einlesen ohne abzuspeichern

Genesis-Online liefert csv- und ffcsv-Dateien gezippt aus, wobei jedes ZIP-Archiv genau eine csv-Datei enthält. Beispielhaft wird hier die direkte Verarbeitung aus dem Arbeitsspeicher gezeigt:

In [57]:
```
def csvUnZip(response):
    filebytes = io.BytesIO(response.content)
    zipFile = zipfile.ZipFile(filebytes)
    csvFile = zipFile.open(zipFile.namelist()[0])
    return csvFile
```

### Parameter zum Einlesen für die deutschsprachige Variante der ffcsv-Dateien

In [58]:
```
def readCsv(csvFile):
    df = pd.read_csv(csvFile, delimiter=';', decimal=",", na_values=["...",".","-","/","x"])
    return df
```

### Zusammenführen der beiden tablefile Antworten

In [70]:
```
df_combined = pd.concat([readCsv(csvUnZip(response1)),readCsv(csvUnZip(response2))])\
                .sort_values(by=["time","1_variable_attribute_code","3_variable_attribute_code"])

df_combined[["statistics_label","time","1_variable_attribute_label","3_variable_attribute_label","value"
```

Out[70]:

| | statistics_label | time | 1_variable_attribute_label | 3_variable_attribute_label | value |
|---|---|---|---|---|---|
| **21299** | Verbraucherpreisindex für Deutschland | 2020 | Januar | Reis | 99.6 |
| **29259** | Verbraucherpreisindex für Deutschland | 2020 | Januar | Reiszubereitung | 100.0 |
| **5864** | Verbraucherpreisindex für Deutschland | 2020 | Januar | Weizenmehl | 100.9 |
| **15401** | Verbraucherpreisindex für Deutschland | 2020 | Januar | Grieß, Roggenmehl oder Ähnliches | 101.3 |
| **1352** | Verbraucherpreisindex für Deutschland | 2020 | Januar | Weißbrot | 101.0 |
| **...** | ... | ... | ... | ... | ... |
| **5042** | Verbraucherpreisindex für Deutschland | 2025 | Februar | Passgebühr oder Ähnliches | 120.7 |
| **4515** | Verbraucherpreisindex für Deutschland | 2025 | Februar | Rechtsanwaltsgebühr oder Notargebühr | 117.4 |
| **3584** | Verbraucherpreisindex für Deutschland | 2025 | Februar | Bestattungsleistungen und Friedhofsgebühr | 120.8 |
| **1652** | Verbraucherpreisindex für Deutschland | 2025 | Februar | Kleinanzeige in einer Zeitung | 119.2 |
| **8143** | Verbraucherpreisindex für Deutschland | 2025 | Februar | Kurtaxe | 107.9 |

42594 rows × 5 columns

## Auftrag erstellen und abholen

Bei sehr großen Tabellen ist der Abruf über eine Warteschlange (Batch) erforderlich. Hierzu ist `data/tablefile` mit dem Parameter `job=true` aufzurufen. Nach einigen Minuten kann über `catalogue/results` der Bearbeitungsstand der angeforderten Ergebnistabelle abgefragt und schließlich mit `data/resultfile` heruntergeladen werden.

## Datenquader recherchieren

Über den `Catalogue` Service kann mit der `cubes` Methode der gewünschte Quader-Code recherchiert werden, der für den Download erforderlich ist.
Im folgenden Beispiel wird eine Liste der Quader aus der Preisstatistik ( `61*` ) angefordert.

In [41]:
```
responseCatalogue = requests.post(BASE_URL + 'catalogue/cubes',
    headers = headers,
    data = {
    'selection': '61*',
    'pagelength': 250,
    'language': langPref
    })
```

In [42]:
```
cubeList = responseCatalogue.json()["List"]
len(cubeList)
```

Out[42]: 247

Der Übersichtlichkeit halber filtern wir die 247 Einträge umfassende Liste mit der Suche nach spezifischen Inhalten

In [43]:
```
[x for x in cubeList if "Betriebsmittel" in x["Content"]]
```

Out[43]:
```
[{'Code': '61221BJ001',
  'Content': 'Index der Einkaufspreise landwirt. Betriebsmittel, Index der Einkaufspreise landwirt. Betriebsmittel, Deutschland insgesamt, Landwirtschaftliche Betriebsmittel, Jahr',
  'State': 'vollständig mit Werten',
  'Time': '1961-2024',
  'LatestUpdate': '12.12.2024 08:01:02h',
  'Information': 'false'},
 {'Code': '61221BJ002',
  'Content': 'Index der Einkaufspreise landwirt. Betriebsmittel, Index der Einkaufspreise landwirt. Betriebsmittel, Deutschland insgesamt, Landwirtschaftliche Betriebsmittel, Wirtschaftsjahr',
  'State': 'vollständig mit Werten',
  'Time': '1938/39-2023/24',
  'LatestUpdate': '12.09.2024 08:01:01h',
  'Information': 'false'},
 {'Code': '61221BM001',
  'Content': 'Index der Einkaufspreise landwirt. Betriebsmittel, Index der Einkaufspreise landwirt. Betriebsmittel, Deutschland insgesamt, Landwirtschaftliche Betriebsmittel, Stichmonat',
  'State': 'vollständig mit Werten',
  'Time': '01/1968-01/2025',
  'LatestUpdate': '13.03.2025 18:05:42h',
  'Information': 'false'}]
```

## Datenquader herunterladen

In [44]:
```
responseCube = requests.post(BASE_URL + 'data/cubefile',
    headers = headers,
    data = {
    'name': '61221BJ002',
    'language': langPref
    })
```

## Speichern der Datei

In [46]:
```
with open("cube.txt", "wb") as f:
    f.write(responseCube.content)
```