24.6.9.2 Projekt Wetter-Daten

Nach *https://de.wikipedia.org/wiki/Openweathermap* ist OpenWeatherMap ein Online-Dienst, der eine programmiersprachen-unabhängige und frei nutzbare Schnittstelle (API) für Wetterdaten bereitstellt. Sie müssen aber über https://home.openweathermap.org/users/sign_up einen kostenfreien API-Key anfordern, um auf diese Wetterdaten zugreifen zu können. Wichtig in Bezug auf das vorgestellte Projekt ist Tatsache, dass alle Wetterdaten im *JSON-Format* (Standard) bezogen werden können. Als eine Einschränkung für die kostenlose API gilt: Die Wetterdaten werden nur alle zwei Stunden aktualisiert.

Für den Aufruf der Wetterdaten für den Ort *Osterburg* in Deutschland wurden diese Werte verwendet:

```
Ort:  q={city name} → q=Osterburg
Ort und Landeskenner: q={city name},{country code} → q=Osterburg,de
Daten-Format: mode={ xml | html } JSON ist Standard
Sprache für kurze Wetterbeschreibung: lang={Landeskenner} → lang=de
Temperatur-Skala: units={ metric (°C) | imperial (F) } Standard ist Kelvin → units=metric
API-Key: APIID={ API-Key } → APIID=YourAPIKEY
```

```
URL:  http://api.openweathermap.org/data/2.5/weather?q=Osterburg,de&lang=de&units=metric&APPID=yorAPIKEY
```

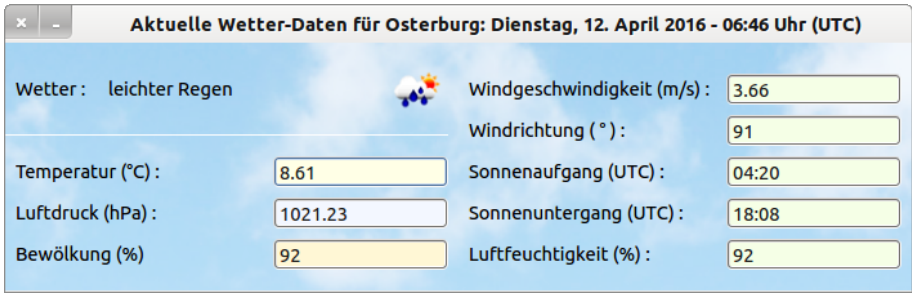

Abbildung 24.6.9.2.1: GUI Wetterdaten

Im Projekt müssen folgende Aufgaben bearbeitet werden:

- Zuerst sind die Wetter-Daten als JSON-Text vom Daten-Server von openweathermap.org/ auf den heimischen PC zu laden und als Text in geeigneter Weise (temporär) zu speichern. Für den Download können Sie das Programm 'wget' und die Instruktionen SHELL oder EXEC oder den HTTP-Client (gb.net.curl) einsetzen, dem der Vorzug gegeben wird. Für den Download sind die folgenden Parameter bereit zu stellen: ein API-Key von openweathermap.org , Ort und Landes-Kenner.
- Dann ist der JSON-Text mit Hilfe der Methode JSON.Decode(JSONString As String [, UseNull As Boolean]) zu dekodieren. Das Ergebnis wird – in Abhängigkeit vom optionalen Parameter UseNull – in einer Variablen vom Typ Collection oder JSON-Collection gespeichert.
- Anschließend sind die Werte der einzelnen Wetter-Daten mit Hilfe der im JSON-Text angegebenen Keys aus der Collection zu ermitteln. Entweder werden die Wetter-Daten sofort in geeigneten Steuerelementen angezeigt → Abbildung 24.6.9.2.1 oder in Variablen abgespeichert. Das Speichern ist dann angebracht, wenn die Daten noch konvertiert und formatiert werden müssen, bevor sie weiterverarbeitet oder angezeigt werden. Das trifft vor allem auf Angaben von Datum und Zeit zu.
- Da in den Wetter-Daten neben der Kurzbeschreibung des Wetters (in deutscher Sprache bei entsprechender Vorgabe für die Sprache) auch der *Bezeichner* für ein passendes Wetter-Symbol angegeben wird, ist ein Download der Symbol-Bild-Datei (png-Format) mit einem weiteren HTTP-Client zu planen. Als (optionale) Parameter für die Methode HTTPClient.([Headers As String[], TargetFile As String]) sind ein leeres String-Array und der *Datei-Pfad* für die Symbol-Bild-Datei zu übergeben.

Der Quelltext wird vollständig angegeben und anschließend kommentiert:

```
[1]  ' Gambas class file
[2]
[3]  Public sJSONData As String
[4]  Public sAPIKey As String
[5]
[6]  Public jCollection As JSONCollection
[7]  Public cWind As Collection
```

```
[8]  Public cClouds As Collection
[9]  Public cMain As Collection
[10] Public cSys As Collection
[11] Public aWeather As Variant[]
[12] Public iTimeStampGMT As Integer
[13] Public sCity As String
[14]
[15] Const _API_KEY As String = "YOUR_API_KEY"
[16] Const _CITY As String = "YOUR_CITY"
[17] Const _COUNTRY As String = "YOUR_COUNTRY"
[18]
[19] Public Sub Form_Open()
[20]
[21]   Dim sFormat, sMessage As String
[22]
[23]   FMain.Center()
[24]   FMain.Resizable = False
[25]   FMain.Title = ("Current weather data")
[26]
[27]   sJSONData = GetWeatherData(_CITY, _COUNTRY, _API_KEY)
[28]
[29] ' Print sJSONData
[30] ' Shell "echo '" & sJSONData & "' | python -m json.tool" To sFormat
[31] ' Print sFormat
[32]
[33]   jCollection = JSON.Decode(sJSONData, True)
[34]
[35]   If jCollection["cod"] = 401 Then
[36]      sMessage = "Sie müssen einen (kostenlosen) API-Key von<br>"
[37]      sMessage &= "<b>https://home.openweathermap.org/users/sign_up</b>"
[38]      sMessage &= "<br>anfordern, um die Wetter-Daten auslesen zu können!"
[39]      Message.Error(sMessage)
[40]      FMain.Close()
[41]   Else
[42]       ShowWeatherData()
[43]   Endif
[44]
[45] End ' Form_Open()
[46]
[47] '-------------------------------------------------------------------------------------------------------
[48]
[49] Private Sub ShowWeatherData()
[50]
[51]   Dim iTimeStampUTC As Integer
[52]   Dim sCityName, sDate, sIcon, sDay, sDateDMY, sTime As String
[53]   Dim dDate As Date
[54]
[55]   sCityName = jCollection["name"]
[56]
[57]   iTimeStampUTC = jCollection["dt"]
[58]   dDate = TimeStampToDate(iTimeStampUTC)
[59]   sDay = Format$(dDate, "dddd")
[60]   sDateDMY = Format$(dDate, "d. mmmm yyyy")
[61]   sTime = Format$(dDate, "hh:nn") & " Uhr (UTC)"
[62]   sDate = sDay & ", " & sDateDMY & " - " & sTime
[63]   FMain.Text = ("Current weather data") & (" for ") & sCityName & ": " & sDate
[64]
[65]   cWind = jCollection["wind"]
[66]   txbWindSpeed.Text = cWind["speed"]
[67]   txbWindDirection.Text = Round(cWind["deg"], 0)
[68]  '-----------------------------------------------
[69] ' cMain = jCollection["main"]
[70] ' txbTemperature.Text = cMain["temp"]
[71] ' txbPressure.Text = cMain["pressure"]
[72] ' txbHumidity.Text = cMain["humidity"]
[73]   txbTemperature.Text = jCollection["main"]["temp"]
[74]   txbPressure.Text = jCollection["main"]["pressure"]
[75]   txbHumidity.Text = jCollection["main"]["humidity"]
[76] '-----------------------------------------------------
[77]   cClouds = jCollection["clouds"]
[78]   txbCloudiness.Text = cClouds["all"]
[79]  '-----------------------------------
[80]   cSys = jCollection["sys"]
[81]   txbSunRise.Text = Format(TimeStampToDate(cSys["sunrise"]), "hh:nn")
[82]   txbSunSet.Text = Format(TimeStampToDate(cSys["sunset"]), "hh:nn")
[83]  '-------------------------------------------------------------------
[84]   aWeather = jCollection["weather"]
[85]   lblDescription.Text = aWeather[0]["description"]
[86]   sIcon = aWeather[0]["icon"]
[87] ' lblDescription.Text = jCollection["weather"][0]["description"]
[88] ' sIcon = jCollection["weather"][0]["icon"]
[89]   GetWeatherIcon(sIcon)
[90]   pboxWeather.Picture = Picture[Application.Path &/ "Icon/weathericon.png"]
[91]
```

```
[92] End ' ShowWeatherData()
[93]
[94] Private Function GetWeatherData(City As String, Country As String, APIKey As String) As String
[95]
[96]   Dim sURL As String
[97]   Dim hHTTPClient As HttpClient
[98]
[99]   sURL = "http://api.openweathermap.org/data/2.5/weather"
[100]     sURL &= "?q=" & City & "," & Country
[101]   sURL &= "&lang=" & Country
[102]   sURL &= "&units=metric"
[103]   sURL &= "&APPID=" & APIKey
[104]
[105]   hHTTPClient = New HttpClient As "hHTTPClient"
[106]   hHttpClient.URL = sURL
[107]   hHTTPClient.Async = False
[108]   hHTTPClient.Timeout = 10
[109]   Try hHTTPClient.Get()
[110]
[111]   Return hHTTPClient.ReadLine()
[112]
[113] End ' GetData(City As String, Country As String, APIKey)
[114]
[115] Private Sub GetWeatherIcon(Icon As String)
[116]
[117]   Dim hHTTPClient As HttpClient
[118]
[119]   hHTTPClient = New HttpClient As "hHTTPClient"
[120]   hHttpClient.URL = "http://openweathermap.org/img/w/" & Icon & ".png"
[121]   hHTTPClient.Async = False
[122]   hHTTPClient.Timeout = 10
[123]   Try hHTTPClient.Get([], Application.Path &/ "Icon/weathericon.png")
[124]
[125] End ' GetWeatherIcon(...)
[126]
[127] Private Function TimeStampToDate(TimeStamp As Integer) As Date
[128]   Return DateAdd(CDate("1/1/1970"), gb.Second, TimeStamp)
[129] End ' TimeStampToDate
```

Kommentar:

- Für die Bearbeitung der ersten Aufgabe wird die Funktion GetData(...) in den Zeilen 94 bis 113 mit den notwendigen Argumenten eingesetzt. Die Argumente werden über drei Konstanten in den Zeilen 15 bis 17 definiert. Der JSON-Text wird in der Zeile 27 in der Variablen sJSONData vom Typ String abgespeichert.
- Zur Kontrolle können Sie in den Zeilen 29 bis 31 den JSON-Text in der Konsole der Gambas-IDE ausgeben. Wesentlich besser lesbar wird der JSON-Text, wenn Sie diesen formatiert ausgeben.
- Die Bearbeitung der zweiten Aufgabe erfordert das Dekodieren des JSON-Textes. Das übernimmt die Methode JSON.Decode(...). Aus Daten im JSON-Format werden Daten vom Gambas-Typ Variant – hier speziell vom Typ JSONCollection.

Analysiert man den folgenden JSON-Text

```
{"coord":{"lon":11.77,"lat":52.78},"weather":[{"id":500,"main":"Rain","description":"leichter
Regen","icon":"10d"}],"base":"cmc stations","main":{"temp":8.61,"pressure":1021.23,"humidity":92,
"temp_min":8.61,"temp_max":8.61,"sea_level":1025.91,"grnd_level": 1021.23},"wind":{"speed":3.66,
"deg":90.505},"rain":{"3h":0.84},"clouds":{"all":92},"dt":1460443601,"sys": {"message":0.0036,
"country":"DE","sunrise":1460434806,"sunset":1460484485},"id":2856639,"name":"Osterburg","cod":200}
```

dann erkennt man für die einzelnen Keys in der JSONCollection neben den Werten der Wetter-Daten auch den Werte-Typ, die hier noch einmal explizit angegeben werden:

```
Anzahl der Elemente in der JSON-Collection: 11
------------------------------------------------------
Schlüssel 1 : "coord" ---> Wert-Typ: JSONCollection
Schlüssel 2 : "weather" ---> Wert-Typ: Variant[]
Schlüssel 3 : "base" ---> Wert-Typ: String
Schlüssel 4 : "main" ---> Wert-Typ: JSONCollection
Schlüssel 5 : "wind" ---> Wert-Typ: JSONCollection
Schlüssel 6 : "clouds" ---> Wert-Typ: JSONCollection
Schlüssel 7 : "dt" ---> Wert-Typ: Integer
Schlüssel 8 : "sys" ---> Wert-Typ: JSONCollection
Schlüssel 9 : "id" ---> Wert-Typ: Integer
Schlüssel 10 : "name" ---> Wert-Typ: String
Schlüssel 11 : "cod" ---> Wert-Typ: Integer
```

Beispiel 1: Ermittlung des Datums und der Zeit, zu der die Wetter-Daten erfasst wurden

Der Schlüssel 7 "dt" steht für DateTime und liefert den Wert zu diesem Schlüssel als Zeit-Stempel (TimeStamp) vom Typ Integer. Wie Sie erkennen können, wurde das vollständige Datum auf eine ganze Zahl abgebildet. In den Zeilen 58 bis 60 erfolgt mit Hilfe der Funktion *TimeStampToDate(..)* die Konvertierung des Zeitstempels in ein Datum (Typ Date), das für die Anzeige in der Fenster-Zeile dreifach anders formatiert wird:

```
iTimeStampUTC = jCollection["dt"]
dDate = TimeStampToDate(iTimeStampUTC)
sDay = Format$(dDate, "dddd")
sDateDMY = Format$(dDate, "d. mmmm yyyy")
sTime = Format$(dDate, "hh:nn") & " Uhr (UTC)"
sDate = sDay & ", " & sDateDMY & " - " & sTime
FMain.Text = ("Current weather data") & (" for ") & sCityName & ": " & sDate
```

Beispiel 2: Ermittlung Temperatur, Luftdruck und Luftfeuchtigkeit

Der Schlüssel 4 "main" liefert als Wert eine Collection mit den Keys "temp", "pressure" und "humidity". Die Werte für Temperatur, Luftdruck und Luftfeuchtigkeit können Sie entweder mit Hilfe der Variablen cMain in den Zeilen 2 bis 4 sofort anzeigen oder Sie verwenden die Zeilen 5 bis 7:

```
[1]  ' cMain = jCollection["main"]
[2]  ' txbTemperature.Text = cMain["temp"]
[3]  ' txbPressure.Text = cMain["pressure"]
[4]  ' txbHumidity.Text = cMain["humidity"]
[5]    txbTemperature.Text = jCollection["main"]["temp"]
[6]    txbPressure.Text = jCollection["main"]["pressure"]
[7]    txbHumidity.Text = jCollection["main"]["humidity"]
```

Beispiel 3: Ermittlung der Wetter-Beschreibung und des Bezeichners für das Wetter-Icon

Der Schlüssel 2 "weather" in der JSONCollection liefert als Wert in diesem Fall ein Variant-Array dessen einziges Element eine Collection ist! Da hier nur die Wetter-Beschreibung und der Bezeichner für das Wetter-Icon interessieren, werden die beiden Keys "description" und "icon" benötigt. Unter Verwendung der Variablen *aWeather* vom Typ Variant[] können Sie die Wetter-Beschreibung sofort anzeigen. Die Zeile 3 in Verbindung mit den Zeilen 6 und 7 lädt die Datei für das Wetter-Icon und zeigt das Wetter-Icon unmittelbar in einer Picture-Box an. Eine Alternative für die Zeilen 1 bis 3 – unter Verzicht auf eine Variable für das Variant-Array – finden Sie in den Zeilen 4 und 5:

```
[1]    aWeather = jCollection["weather"]
[2]    lblDescription.Text = aWeather[0]["description"]
[3]    sIcon = aWeather[0]["icon"]
[4]  ' lblDescription.Text = jCollection["weather"][0]["description"]
[5]  ' sIcon = jCollection["weather"][0]["icon"]
[6]    GetWeatherIcon(sIcon)
[7]    pboxWeather.Picture = Picture[Application.Path &/ "Icon/weathericon.png"]
```

Hier sehen Sie den gut lesbaren JSON-Text, weil er formatiert ausgegeben wird:

```
{
    "base": "cmc stations",
    "clouds": {
        "all": 92
    },
    "cod": 200,
    "coord": {
        "lat": 52.78,
        "lon": 11.77
    },
    "dt": 1460443601,
    "id": 2856639,
    "main": {
        "grnd_level": 1021.23,
        "humidity": 92,
        "pressure": 1021.23,
        "sea_level": 1025.91,
        "temp": 8.61,
        "temp_max": 8.61,
```

```
        "temp_min": 8.61
    },
    "name": "Osterburg",
    "rain": {
        "3h": 0.84
    },
    "sys": {
        "country": "DE",
        "message": 0.0036,
        "sunrise": 1460434806,
        "sunset": 1460484485
    },
    "weather": [
        {
            "description": "leichter Regen",
            "icon": "10d",
            "id": 500,
            "main": "Rain"
        }
    ],
    "wind": {
        "deg": 90.505,
        "speed": 3.66
    }
}
```

Wenn in der Wetter-Beschreibung zum Beispiel Umlaute enthalten sind, so werden diese konvertiert:

```
"weather": [
  {
    "description": "\u00fcberwiegend bew\u00f6lkt",
    "icon": "04d",
    "id": 803,
    "main": "Clouds"
  }
]
```

Auch die Fehlermeldungen erhalten Sie im JSON-Format – wie hier bei einem fehlendem API-Key:

```
{
    "cod": 401,
    "message": "Invalid API key. Please see http://openweathermap.org/faq#error401 for more info."
}
```

Dieser Fehler wird auch im Projekt in den Zeilen 35 bis 43 abgefangen:

```
If jCollection["cod"] = 401 Then
   sMessage = "Sie müssen einen (kostenlosen) API-Key von<br>"
   sMessage &= "<b>https://home.openweathermap.org/users/sign_up</b>"
   sMessage &= "<br>anfordern, um die Wetter-Daten auslesen zu können!"
   Message.Error(sMessage)
   FMain.Close()
Else
   ShowWeatherData()
Endif
```

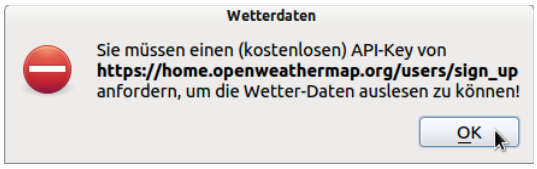

Abbildung 24.6.9.2.2: Fehlermeldung

Das vorgestellte Projekt ist eine Adaption eines Projektes von Steve Dee, das er auf seiner Website http://captainbodgit.blogspot.de vorgestellt hat.