Schnittstellen der MiData

Übersicht

Um Daten aus der MiData zu beziehen, gibt es grundsätzlich zwei Möglichkeiten:

Service-Tokens

Service Accounts ermöglichen es, für eine externe Applikation einen eigenen Account mit bestimmten Berechtigungen zu erstellen, mit dem sie dann die JSON-Schnittstelle nutzen kann. Service Accounts werden pro Ebene von einer berechtigten Person erstellt und bleiben auch bestehen, wenn diese Person die Gruppe verlässt oder gelöscht wird.

Spezifikation (Englisch)

OAuth

Die MiData ist ebenfalls ist ein OAuth 2.0 Anbieter, das heisst dass eine externe Applikation Benutzer via MiData authentifizieren kann. Du kennst dieses Prinzip vielleicht bereits, wenn du einmal "Anmelden mit Facebook" oder "Anmelden mit Google" verwendet hast. Die externe Applikation kann danach Informationen über den Benutzer aus der MiData abfragen wenn der Benutzer dies selber freigibt. Die OAuth-Anmeldung ermöglicht es auch, dass die externe Applikation die JSON-Schnittstellen im Namen des Users nutzen kann. Die Applikation hat dabei dieselben Berechtigungen wie der Benutzer.

Spezifikation (Englisch)

Wenn du die MiData als OAuth-Provider für eine deiner Applikationen verwenden willst, musst du den OAuth-Antrag ausfüllen. Der IT-Support der PBS wird dir einen Zugang auf das Produktivsystem vergeben, insofern alle Vorbedingungen erfüllt sind: Antrag OAuth Applikation

Aktuell: Am 14.09.2021 wurden die persönlichen User Tokens als veraltet (deprecated) markiert. Diese werden also in absehbarer Zeit nicht mehr verfügbar sein. Anwendungen, die bisher mit persönlichen User Tokens betrieben wurden, sind neu per OAuth oder Service-Tokens anzubinden.

Code-Beispiele

Service-Tokens, Beispiel mit R

```r
# Install the required packages as follows:
# install.packages(c("jsonlite", "httr"))

library(jsonlite)
library(httr)

# Script parameters
baseurl <- "https://pbs.puzzle.ch"
token <- "d-Zx8kn-mKWaxXziYs62xVX2HdUdVKnSmLQYpQG-XznkbRD71g"
groupid <- 1

# Get MiData root group
rootElement <- fromJSON(paste(baseurl,"/groups/", groupid, ".json?token=", token, sep=""))
# Filter cantons
cantons = subset(rootElement$linked$groups, group_type == "Kantonalverband")
rownames(cantons) <- seq(length=nrow(cantons))

# Set a flag if the canton has an even id (just for fun)
cantons$even <- F
counter <- 1
for (id in cantons$id) {
       cantons$even[counter] <- ((as.numeric(cantons$id[counter]) %% 2) == 0)
       counter = counter + 1
}

# Print table of cantons
cantons
```

OAuth – hilfreiche Ressourcen

- Hitobito OAuth Flow
- Ruby (Doorkeeper, Devise, Omniauth) Example
- https://oauthdebugger.com
- https://www.oauth.com/playground/