CommonAPI2 Erweiterungen für Mods

Inhaltsverzeichnis

- 1 mod.lua
  - 1.1 modid
  - 1.2 requiredMods
- 2 Endhaltestellen Anzeige (LINE_DESTINATION) nachrüsten
- 3 Gruppierung im Konstruktionsmenü
  - 3.1 Gleise, Straßen oder Konstruktionen
  - 3.2 Modelle .mdl Gruppieren:
- 4 Strassenhaltestellen Erweiterung

Eine Übersicht der CommonAPI2 Erweiterungen:

1 mod.lua

CommonAPI2 erlaubt es Abhängigkeiten und Ladereihenfolgen zu überprüfen. Damit man Steam und lokal installierte Mods mischen kann, braucht es ein paar Erweiterungen im info Block der mod.lua Datei.

1.1 modid

Damit Mods auch aus einer Steam Quelle erkannt werden können, kann hier die originale modid übergeben werden. Bitte die Schreibweise beachten.

Eure `modid` ist gleich des Verzeichnisname, daher [a-zA-Z0-9_-]_[0-9]+, diese kommt in den `info` Block der mod.lua

Code

```
modid = "beispiel_1"
```

1.2 requiredMods

requiredMods Tabelle

Eine Liste mit detaillierten Informationen zu Abhängigkeiten, für jeder benötigte Mod kann optional

`modId`, `steamId` (WorkshopID), `tfnetId` (Webdisk Eintrag), `url` angeben werden.

Es muss mindestens eine `modId` oder `steamId` angeben werden.

CommonAPI2 überprüft die Angabe und gibt eine Warnung in der stdout.txt aus, sollte die Mod nicht geladen sein. Darüber hinaus gibt CommonAPI2 automatisch Download URLs aus

Ob die die Abhängigkeiten vor dem Mod geladen werden müssen (Reihenfolge) kann durch `requiredModsAnyLoadOrder` bestimmt werden,
dieses ist per default auf `false`, d.h. die Ladereihenfolge muss eingehalten werden.

Code: mod.lua

```
info = {

requiredMods = {
 {
        modId = "example_1", -- optional,
    steamId = 0000, -- optional, steam workshop
  tfnetId = 0000, -- optional, id der Webdisk auf transportfever.net
     url = "http://", -- optional, url zum Download der Mod
      minMinorVersion = 0 -- optional, minimale minorVersion
 },
        -- optional, weitere
},

requiredModsAnyLoadOrder = true,   -- optional, ist die Reihenfolge der Mods egal?

},
```

Alles anzeigen

2 Endhaltestellen Anzeige (LINE_DESTINATION) nachrüsten

Sobald `LINE_DESTINATION` eingeschaltet ist, funktioniert in labelList type = "LINE_DESTINATION", ohne CommonAPI2 gibt es einen Ladefehler, daher müsst Ihr eine Code Weiche einbauen:

Siehe dazu auch: https://www.transportfever2.co…ourcetypes:mdl#label_list

In eurer mdl Datei direkt hinter `function data()` einfügen:

Code

```
local labelType = "NEXT_STOP"

if(commonapi~=nil and commonapi.supports and commonapi.supports("LINE_DESTINATION"))then
    labelType
end
```

Danach könnt Ihr in der labelList

`type = "LINE_NAME",` durch `type = labelType,` ersetzten.

Sollte nun der Nutzer mit LINE_DESTINATION spielen, so erhält euer Fahrzeug diese Funktion.

Beachtet: Zurzeit sind Funktionen wie regex oder ähnliches nicht verfügbar!

3 Gruppierung im Konstruktionsmenü

mit CommonAPI2 Funktion improveui

3.1 Gleise, Straßen oder Konstruktionen

Die Gruppierung für Gleise, Straßen oder Konstruktionen funktioniert über den Beschreibungstext:

Es kann sowohl ein Gruppennamen und optional eine Sortierreihenfolge angeben werden in der Form:

`@!<gruppenname>`

oder

`@!<gruppenname>-<reihenfolge>`

Oder als LUA Regex ausgedrückt:

`@!([a-zA-Z0-9-_]+[a-zA-Z_])(-?%d?)`

Hier ein Beispiel, wie man weiterhin seine Übersetzung mit Gruppierung Nutzen kann.

Code: mystreet.lua

```
function data()
return {
  desc = _("Meine Strasse %2%.").." @!mystreet-1"
```

3.2 Modelle .mdl Gruppieren:

Für mdls kann uiCommonAPI im metadata Block nutzen.

`groupName` muss als string gesetzt werden, `groupValue` ist ein optionaler Integer Wert, der die Sortierung beeinflusst.

Code: irgendwas.mdl

```
function data()
return {
    metadata
        groupValue = 123
    }
}
```

Metadaten kann im Modfifer geändert werden, es ist nich möglich dieses in einer postRunFn zu ändern (CommonAPI2 hat noch keine öffentliche Schnittstelle dafür)

Für Modelle kann man auch alternativ wieder den Beschreibungstext nutzen:

Code: irgendwas.mdl

```
metadata = {
    description
        description = _("Meine test mdl").."
}
```

4 Strassenhaltestellen Erweiterung

CommonAPI2 Funktion `flexStreets` und `newevents` müssen aktiviert sein.

Bus & Tramhaltestellen .mdl Dateien können um den `streetTerminalCommonAPI` im `metadata` Block erweitert werden:

Alle Werte sind optional und müssen daher nicht unbedingt angeben werden:

- moreCapacity = <positiver integer> damit können mehr Passagiere an einer Haltestelle Warten (zusätzlicher "Lagerplatz" wie bei Konstruktionsgebäude)
- upright = <true/false> Soll das Modell immer senkrecht gebaut werden. Bei `false` wird das Modell an der Strasse ausgerichtet, sprich auf Steigungen gekippt.
  (Default ist true, und man sollte upright = true, daher nicht verwenden!)
- vehicleLane = <integer> welche LaneId ein Fahrzeug hält. Postive Werte = gerechnet vom Bürgersteig, negative Werte = von der Mitte der Straße aus gerechnet.
  (Nur Benutzen, wenn man diese Funktion auch braucht!)

Beispiele:

Code: haltestelle_mehrkapazitaet.mdl

```
metadata                                    =                                    {
                              streetTerminalCommonAPI                              =
 }
```

Code: haltestelle_StrassenMitte_mehrKapazitaet.mdl

```
metadata                                    =                                    {
                               streetTerminalCommonAPI                               =


 },
```


https://dev.transportfever.net/lexicon/entry/377-commonapi2-erweiterungen-f%C3%BCr-mods/