

# Piste-skeemat

## YLEISTÄ:

Tietokantapiste on tietyltä kannalta ajateltuna Lua taulukko - lua tyyppinä "table". Koska lua-tilukko ei sinänsä aseta mitään rajoituksia sille, millaista tietoa siihen voidaan tallentaa, SLC enginen pistetietokanta toteuttaa pisteille skeema tarkastelun. Aina kun pisteen johonkin kenttään kirjoitetaan arvo, kyseistä arvoa verrataan pisteen skeemaan, ja mikäli kirjoitettava arvo ei vastaa skeeman määrittystä - ja mikäli sitä ei voida konvertoida oikeaan muotoon, kirjoitus epäonnistuu.

Tietokantapisteiden skeemat seuraavat pääpiirteittäin "JSON schemas" määrittystä, mutta hieman yksinkertaistettuna ja toisaalta, sisältäen taas joitakin laajennoksia. Pistetietokanta asettaa jo lähtökohtaisesti kuitenkin joitakin lisärajoituksia; Koska skeemalla kuvataan pistetietokantaobjektia, tulee ylimmän tason tyyppin olla sen vuoksi aina "object" - skeema jossa näin ei ole, hylätään.

## SKEEMAN, ELI PISTETYYPPIN LUOMINEN:

Uusi pistetyyppi - eli skeema, luodaan Data -kirjaston kutsulla createSchema.

```
Data.createSchema ( strSchemaName, schemaModel )
```

### Argumentit:

strSchemaName Merkkijono joka tulee uuden skeeman nimeksi. Voi sisältää merkkejä a-z, A-Z ja \_

### Paluuarvo:

Kutsu palauttaa arvon true jos onnistui, tai nil jos epäonnistui.

Jos haluaisi kuvata vaikkapa muuttujan, jonka täytyy sisältää kokonaisluku, väliltä 0 .. 1, olisi skeema esimerkiksi seuraavanlainen:

```
{type="int", default=0, minimum=0, maximum=1}
```

Mutta kuten ylempänä jo sanottiin, täytyy tietokantapisteen olla aina ylimmällä tasolla object tyyppinen - se on vaatimus, jonka ohjelma tarkistaa. object -tyyppi tarkoittaa taulukkoa, joka sisältää rivejä, joiden nimet ovat merkkijonoja. Kun asiaa miettii, juuri sellaisiahan datapisteet ovat; taulukko, joka sisältää rivejä, joilla on nimiä kuten "pv", tai "description".

Näitä tyyppejä "object", "array", "int", "real", "string", "boolean" kutsutaan datatyypeiksi. Jokaisella datatyyppillä on lista lisävaatimuksia, joita voidaan määrätä kyseisen tyyppiselle tiedolle. Esimerkiksi object tyyppiselle datalle

voidaan esittää vaatimuksia siitä, millaisia rivejä sen on pakko sisältää. Array tyyppiselle, eli taulukolle (tai joskus kutsutaan listaksi) voidaan määrätä minimi tai maksimi määrä rivejä, ja jokaiselle riville jokin pakollinen muotovaatimus (esimerkiksi että rivit ovat kokonaislukuja).

Lua -kielen kannalta skeema on vain tietyllä tavalla muotoiltu taulukko. Kun kyseinen taulukko annetaan parametrina `Data.createSchema ()` -funktiolle, kyseinen funktio laittaa tuon annetus taulukon muistiin, ja järjestelmä vertaa niitä datapisteitä siihen, joiden on ilmoitettu olevan tuota tyyppiä.

Esimerkki: BI-piste:

```
Data.createSchema ("BI", {
  type="object",
  required={"class", "description", "pv", "priority", "inactiveText", "activeText", "faultStatus",
    "outOfService", "reliability", "description", "commStatus", "commTxt", "dataSource"},
  additionalProperties=true,
  properties={
    class={type="text", default="BI", fixed="BI"},
    description={type="text", default="BI object"},
    pv={type="int", default=0, minimum=0, maximum=1},
    priority={type="int", default=16, minimum=1, maximum=16},
    inactiveText={type="text", default="Off"},
    activeText={type="text", default="On"},
    faultStatus={type="integer", default=0, minimum=0},
    outOfService={type="integer", default=0, minimum=0, maximum=1},
    reliability={type="integer", default=0, minimum=0},
    commStatus={type="integer", default=0},
    commTxt={type="text", default=""},
    dataSource={type="text", default=""}
  }
})
```

#### DATATYYPIT JA NIIDEN OMINAISUUDET:

**boolean** totuusarvo joka voi saada arvon true tai false. Yksinkertaisin skeemojen tukema tietotyyppi.

"fixed" tällä määreellä voidaan pakottaa muuttujan arvo halutuksi.

"default" oletusarvo kun muuttuja luodaan.

Aliakset: bool

**int** Kokonaisluku. ARM arkkitehtuurilla tämä datatyyppi muutetaan 32 bittiseksi kokonaisluvuksi.

"fixed" tällä määreellä voidaan pakottaa muuttujan arvo halutuksi.

"default" oletusarvo kun muuttuja luodaan.

"minimum" pienin numeraalinen arvo jonka muuttuja voi saada

"maximum" suurin numeraalinen arvo jonka muuttuja voi saada

"enumerated" lista arvoista joihin muuttuja voidaan asettaa.

Aliakset: integer

number Reaaliluku, jota kuvataan 64 bittisellä liukuluvulla. Lua kielen natiivi numerotyyppi.

"fixed" tällä määreellä voidaan pakottaa muuttujan arvo halutuksi.

"default" oletusarvo kun muuttuja luodaan.

"minimum" pienin numeraalinen arvo jonka muuttuja voi saada

"maximum" suurin numeraalinen arvo jonka muuttuja voi saada

"enumerated" lista arvoista joihin muuttuja voidaan asettaa.

Aliakset: real, float

array Lista tyyppinen juoksevasti kokonaisluvuilla indeksoitu taulukko, jonka rivit on nimetty juoksevasti kokonaisluvuilla.

HUOM! Vaikka lua -kieli ei varsinaisesti erottelekaan array ja object tyyppisiä taulukoita, on myös lua kielessä nopeampaa iteroita taulukon solut lävitse ipair() kuin pair() kutsulla, eli listat ovat nopeampia käsitellä kuin hash taulukot.

"itemNamees" Tällä listalla voidaan antaa taulukon riveille näyttönimet.

"minItems" minimi-määrä rivejä taulukossa.

"maxItems" maksimi-määrä rivejä taulukossa.

"items" on kutakin riviä kuvaava skeema - jokainen taulukon rivi joutuu seuraamaan tätä mallia!

object Taulukko, jonka rivit on indeksoitu merkkijonoilla, eli hash taulukko.

Huom! Tämä datatyyppi ei ole täysin natiivi lua -kielen kannalta, sillä lua -taulukko voisi sisältää sekä numeraalisesti indeksoituja rivejä, että merkkijonoilla indeksoituja. Eli se ei tee eroa object ja array tyyppien välillä. Tämä käytäntö periytyy Javascript kielestä, ja JSON notaatiosta.

"required" kenttä on lista merkkijonoja, joka määrää mitä rivejä objektin on pakko sisältää

"additionalProperties" on true/false arvo, joka ilmoittaa voiko objekti sisältää muitakin kuin 'properties' taulukossa kuvattuja rivejä.

"properties" on hash-taulukko jossa on annettu taulukon jokaisen

rivin oma skeema, joka siis kuvaa kyseiseen riviin sovellettavaa skeemaa.

Revision #1

Created 30 May 2022 11:03:57 by Severi Hiltunen

Updated 10 June 2022 11:24:05 by Severi Hiltunen