Memo

Aan
Leden DD-werkgroep

Datum	**Kenmerk**	**Aantal pagina's**
7 oktober 2021	11207342-DSC-19-v0.9	3

Van	**Doorkiesnummer**	**E-mail**
Stef Hummel	+31(0)6 1019 8112	Stef.Hummel@deltares.nl

Onderwerp
Bespreekverslag web conference over DD filter mechanisme d.d. 24 augustus

---

Aanwezig:	
Bram de Graaf	HydroLogic
Carsten Byrman	Nelen & Schuurmans
Flip Dirksen	RWS
Geri Wolters	EcoSys
Jeroen Gerrits	Vortech
Jurgen Boerboom	RWS/IBM
Stef Hummel	Deltares
Verhinderd:	
Bart Thonus	HKV
Dirk Schwanenberg	Kisters

Aanpak DD filtering mechanisme

Tijdens de discussie kwamen een aantal langs, die zich laten samenvatten tot de volgende vragen:

- Huidige filter-syntax blijven gebruiken of over naar GraphQL?
- Al of niet POST gaan ondersteunen?
- Wel of niet altijd quotes rond strings in een query?
- Voorgedefinieerde filters blijven gebruiken?

Filter-syntax

Casper suggereerde om te kijken naar GraphQL, een veelgebruikt query-mechanisme in API's. Bestuderen en bespreken daarvan leidde tot de conclusie dat het veel meer biedt dan we nodig hebben. Gebruiken van GraphQL zou dan leiden tot onnodige complexe formuleringen van de query. Om zeker te weten dat we ons daarin niet vergissen gaat Jeroen er wat beter naar kijken samen met collega Werner Kramer, die er ervaring mee heeft (**Actie**: Jeroen). *(Opmerking ten tijde van het uitwerken van de notulen: dit is ondertussen gebeurd, met inderdaad dezelfde conclusie, zie issue 38. Ook blijkt GraphQL met POST requests te werken, daar waar tot nu de DD-API volledig GET-gebaseerd is.)*

POST

De DD-API is GET-gebaseerd, en dat wordt als 'correct' ervaren: een systeem wordt bevraagd, en wordt niets aan geleverd. Maar een GET-url heeft een beperkte lengte (zie b.v.

stackOverflow bericht), en dat kan tot problemen leiden bij een vraag om meerdere tijdseries waarbij een uitgebreide filter-specificatie wordt meegegeven, b.v. 'IN(…lange lijst stations…). Vraag is of we daarom niet alsnog ook POST moeten gaan ondersteunen.
Actie: Geri zal aan de hand van voorbeelden beargumenteren waarom POST nodig is.
(Opmerking ten tijde van het uitwerken van de notulen: dit is ondertussen gebeurd, zie bijlage onderdaan dit verslag.)

Quotes rond strings

In een DD-API filter-query moeten momenteel rond een string (b.v. de naam van een station) altijd quotes worden geplaatst. Casper geeft aan dat het merendeel van de API's die hij kent zich redden zonder quotes.
Hoewel we hier geen al te zwaar onderwerp van maken willen we toch onderzoeken of we in inderdaad altijd quotes willen, of dat ze b.v. alleen vereist worden in geval van een naam met spaties.
Actie: Carsten zal de afwegingen in kaart brengen.

Voorgedefinieerde filters

We kennen de voorgedefinieerde filters *locationFilter*, *observationTypeFilter* en *timeseriesFilter*. Vraag is of we dit zo willen houden (en welke er dan bij zouden moeten), of dat we één generieke *filter=…* parameter willlen kennen.
Actie: Stef zal de argumentatie voor de huidige filteropzet opschrijven en rondmailen.

Acties:

Wie	***Omschrijving***
Jeroen	GraphQL onderzoeken (gedaan, zie issue 38)
Geri	Noodzaak POST motiveren (gedaan, zie bijlage)
Carsten	Afwegingen al of niet altijd quotes.in kaart brengen
Stef	argumentatie voor de huidige filteropzet opschrijven en rondmailen

Bijlage: Waarom POST?

(Geri Wolters)

GETs hebben een beperkte lengte van de URL die kan worden verstuurd. Een veilige marge is 2000.

Hieronder een aantal scenario's waarbij dit een probleem is.

Voorbeeld 1, DD-API: Voor HHN hebben we voor het Crisisportaal peilgebieden moeten gebruiken om te achterhalen welke meetobjecten gebruikt moeten worden. Die waren in de vorm van WKT. De grootte daarvan bedroeg meer dan 16MB aan tekst.
Dit kan niet via URLs of browsers worden gestuurd, en lang niet alle platformen kunnen die URL lengtes aan.
Dit kan niet gesplitst worden, want je zult later alsnog moeten gaan filteren op delen van de peilgebieden.

Voorbeeld 2: DD-API, DD-ECO-API chemie. Selecteer metingen voor alle medicijnresten. Dit zijn er ook honderden.
Deze zouden echter wel worden gesplitst, echter eist dit meer van de opvrager. Die moet namelijk dan de data consolideren, wat voor browser-gebruikers niet te doen is.
Bijkomend probleem is dat er geen officiële online-lijsten zijn met deze stoffen.

Voorbeeld 3: DD-ECO-API AquaDesk. Er was een verzoek om metingen op te halen van alle taxa die vallen onder de 11.000 leden tellende groep Polychaeta (borstelwormen). Dit kan uiteraard niet via een URL.
Voor AquaDesk is er daarom een speciale operator gemaakt, genaamd :tree: (voor tree of life) te maken die de selectie doet, door zelfstandig in de relatie tussen taxa te doorzoeken. Dit kan waarschijnlijk niet in andere systemen.
Dit zou kunnen worden opgesplitst, maar zou moeten worden opgesplitst in honderden afzonderlijke requests en later gecombineerd.

De resultaten van GETs kunnen worden gecombineerd, uiteraard, echter zal dit voor de gebruiker (vooral die van de browser) een hele moeilijke taak zijn.

Nadelen POST? Een POST wordt niet gecached (behalve wanneer…). Dit lijkt me echter geen groot probleem…

Verder:
Een POST is bedoelt om data te sturen die wordt gebruikt om een resource aan te maken op de server.
Maar kijk een als volgt hierna. De query die je in de body stuurt maken in principe een resource aan op de server (namelijk de zoekcontext). Het verschil is dat het geen resource is die wordt opgeslagen, maar dat is ook nergens gedefinieerd dat dat moet.
Dat POST informatie teruggeeft, is heel normaal, ook voor CRUD operaties.

Ik zie derhalve niet veel bezwaren voor het gebruik van POST.