Webservice api

Andreas De Lille

March 5, 2014

1 Inhoud

In dit document som ik eerst een aantal mogelijkheden op voor de webservice. Bij elke mogelijkheid worden de voor- en nadelen vermeld. De derde methode zal uiteindelijk uitgewerkt worden. Deze methode wordt dus in meer detail uitgeschreven.

2 Eerste uitwerking

Deze eerste uitwerking heeft voor elke test een apart functie. Zo is bijvoorbeeld voor de resultaten van ping op te vragen een functie getPing() die de ping van een testbed terug geeft.

2.1 Functies

Hierbij zijn er veel verschillende functies nodig.

- getPing
- getDuration
- getTestbedName
- getTestDescription
- ..

2.2 Besluit

Hierbij heeft men per functie een pagina die specifiek voor dat deel informatie kan teruggeven. Het nadeel is wel dat er veel verschillende functies zijn die bijna hetzelfde doen. Verder hebben veel van die functies dezelfde parameters. het zou handiger zijn om deze functie meer generisch op te stellen. Gebruikers moeten altijd een andere functie aanroepen. Dit is dus allicht niet het beste idee. Bijgevolg zal ik hier niet verder op ingaan.

3 Tweede uitwerking

De tweede mogelijkheid is om meer generisch te werken. Met een functie getResult() om het resultaat van een test op te halen. We moeten deze functie natuurlijk wel meegeven van welke test we een resultaat willen. Daarnaast zou ik nog 2 functies voor zien getTest en getTestbed die info geven over respectievelijke een test en een testbed.

3.1 Functies

Deze uitwerking heeft maar 3 functies

- get data van een test opvragen.
- getTest Test opvragen.
- getTestbed Testbed opvragen.

3.1.1 Besluit

We zien hier nog altijd een verdeling van functies die eigenlijk niet noodzakelijk is. Waarom zou het opvragen van een commando van een test anders moeten verlopen dan het opvragen van een resultaat? Uiteindelijk is dat ook data die verbonden is met een test. De derde methode werkt dit verder uit.

4 Derde uitwerking

Een derde mogelijkheid is een generische functie. Daaraan moeten we meegeven wat we willen krijgen. Deze methode lijkt de beste uitwerking te zijn en zal dus gebruikt worden.

4.1 Functies

Er zijn een aantal functies. Merk op dat voor al deze functies de request-method van de http-header met get ingevuld zal zijn. Aanpassen van de monitoring informatie zal pas later gedefinieerd worden. Eventueel in een andere webservice of met behulp van een andere functie.

- last : Deze functie geeft het laatste resultaat terug (per test,testbed) dat voldoet aan de opgelegde eisen.
- list : Deze functie geeft een lijst resultaten die voldoen aan de opgegeven parameters. Zo kan deze lijst beperkt worden tot een resultaten die overeenkomen met een bepaalde status.
- average : Deze functie geeft het gemiddelde van de resultaten (per test,testbed) tussen een periode.
- detail : Deze functie geeft een detailweergave terug van een test. Deze detail weergave kan bijvoorbeeld ook logfiles bevatten om debugging te vergemakkelijken.

Merk op de alle functies een lijst met waarden kunnen teruggeven. Het verschil tussen list en de andere is dat list bedoeld is om een reeks waarden te geven die voldoen per test,testbed. De functies 'last' en 'average' zullen voor elke test,testbed een waarde teruggeven.

De functie last kan eenvoudig omgezet in een overeenkomstige list. Toch lijkt deze functie mij handig om snel de actuele statussen te kunnen opvragen.

4.2 Parameters

Hier worden de parameters besproken. Deze parameters zullen het resultaat van de funties hierboven filteren.

- tests : Deze paramater duidt een of meerdere testen aan.
- testbeds : Deze parameter duidt een of meerdere testbeds aan.
- count : Deze parameter geeft het aantal laatste resultaten aan dat teruggegeven moet worden. Tenzij er enkel een from parameter opgegeven is, dan worden de eerste x resultaten startend vanaf de from terug gegeven.
- from : Geeft aan vanaf welke datum er gezocht moet worden.
- to : Geeft aan tot welke datum de resultaten gezocht moeten worden.
- status : Geeft aan welke status de resultaten moeten hebben, bijvoorbeeld fail of succes.
- resultid : Geeft het id van een testresultaat waarvan men een detail weergave wil.

Indien bij tests of bij testbeds de waarde all wordt gegeven, worden respectievelijk alle tests of alle testbeds teruggegeven.

4.3 Return waarden

Er kan gekozen worden tussen xml en json. Het verschil tussen xml en json is dat xml een boomstructuur beschrijft. De json voorstel komt meer overeen met een hashmap. Xml is meer geschikt voor grote geavanceerde structuren. Aangezien het hier om eenvoudige monitoring informatie gaat, is het gebruik van xml af te raden. Verder dient te worden opgemerkt dat een xml-notatie van een object langer is dan de overeenkomstige json. Ook is het parsen van json eenvoudiger. Bijgevolg zal hier dus geen xml, maar json gebruikt worden. Door de mvc opbouw die ik hier zal gebruiken, zal het echter niet moeilijk zijn om xml functionaliteiten te voorzien. Dit kan achteraf eventueel toegevoegd worden.

Merk op dat de huidige situatie niet voorziet in een webservice. De website maakt rechtstreeks verbinding met de databank. Het tussenvoegen van een webservice kan overhead veroorzaken. Door calls te bundelen, zodat een call direct alle resultaten van een testbed teruggeeft, wordt deze overhead tot een minimum beperkt.

Sommige opvragen geven langere antwoorden terug. Zo is de status van een stitching test multivalued. Een stitching test zal proberen om connectie te maken met een testbed. Vervolgens zal hij aanmelden en een testnetwerk opzetten. Eenmaal het testnetwerk opgezet is zal hij proberen om te pingen tussen de verschillende nodes. Een stitching test kan bij elke stap mislopen. Daarom zal er in de status duidelijk vermeld moeten worden welke stappen gelukt zijn en welke niet. Zo kan het zijn dat een stitching test niet kan aanmelden, wat niet verwonderlijk is als hij bv. ook niet kan pingen naar een testbed. De status van een stitching test zal dan ook overeen komen met een lijst van statussen.

De teruggegeven data zal ook generiek gemaakt worden als een soort geneste hash/array waarin alle waarden zitten. Dit geeft het voordeel dat we ons geen zorgen moeten maken over attributen die niet ingevuld zijn, deze worden dan gewoon weggelaten. Het dan ook aan de client om de teruggegeven data om te zetten naar klassen, indien nodig.

4.4 Voorbeelden

Hieronder zal ik een aantal voorbeelden van calls en bijhorende antwoorden geven. Dit is ook enkel beperkt tot opvragingen (GET requests op http niveau).

4.4.1 last?test=all&testbed=all

Deze call zal voor elk testbed het laatste resultaten van elke test teruggeven. Doordat deze call snel de actuele stand van zake teruggeeft, is het makkelijk om via website deze info in een keer op te halen. Dit verkleint de overhead die komt van meerdere calls te moeten afwerken.

```
[ {
  "Testbed": "VirtualWall",
  "results" : [
    "result" : {
      "testName" : "Ping",
      "TestbedName" : "VirtualWall",
      "Value" : "56",
      "Status" : "Good",
      "resultId" : "2227653503"},
    "result" : {
      "testName" : "LoginTest",
      "testbedName" : "VirtualWall",
      "Status" : "Good",
      "resultId" : "2226065734"},
    ...]},{
  "testbed": "testBed1",
  "results" : [
    "result" : {
      "testName" : "Ping",
      "TestbedName" : "testBed1",
      "Value" : "73",
      "Status" : "Warn",
      "resultId" : "2235230823"},
    "result" : {
      "testName" : "LoginTest",
      "testbedName" : "testBed1",
      "Status" : "Good",
      "resultId" : "2226052731"},
    ...]},{
    ...}]
```

De return waarde is een vereenvoudigde weergave van de resultaten. Met behulp van de funtie detail en het id kan een detail weergave van de test opgevraagd worden.

4.4.2 list?testbed=VirtualWall&test=ping&count=10

Deze call geeft voor elk testbed het resultaat van de laatste 10 uitgevoerde pingtesten teruggeven.

```
[ {
  "Testbed": "VirtualWall",
  "results" : [
    "result" : {
      "testName" : "Ping",
      "TestbedName" : "VirtualWall",
      "Value" : "56",
      "Status" : "Good",
      "resultId" : "2227653503"},
    "result" : {
      "testName" : "Ping",
      "TestbedName" : "VirtualWall",
      "Value" : "37",
      "Status" : "Good",
      "resultId" : "2227653423"},

      ... //result 3->10

   ]
}]
```

4.4.3 average?testbed=ALL&test=ping&from=2014-03-1T02:00:00&count=10

Deze functie geeft het gemiddelde van de eerste 10 resultaten van de ping test op alle testbeds uitgevoerd na 1 maart 2014 2uur s'nachts.

4.4.4 detail?resultid=2227653423

Deze call zal een detail geven van het resultaat met id = 2227653423. Dit resultaat bevat naast de timestamp, status en waarde ook logfiles. Dit moet het voor beheerders mogelijk maken om oorzaken van fouten op te sporen.

4.4.5 detail?test=ping&testbed=VirtuallWall&count=10

Deze call zal een gedetailleerde weergave van de laatste 10 ping tests op de virtual wall teruggeven.

4.5 Opmerkingen

- Wordt een type test (bijvoorbeeld een ping test) altijd met dezelfde frequentie uitgevoerd, of moet deze kunnen verschillend van testbed tot testbed? Indien deze niet moet verschillend kunnen we de frequentie bij de test opslaan. Als dit wel moet kunnen moeten we de frequentie per testbed bijhouden wat aanpassingen vereist. Dit komt omdat we dan per testbed een frequentie moeten opslaan en niet meer per test.

- Merk op dat de status telkens opgeslagen wordt in de databank. Voor een simpele pingtest is dit eigenlijk overbodig. Er kan ook op cliënt-side beslist worden dat een ping van 200ms overeen komt met een Warn status. Me zou dus denken dat de status overbodig is. Er zijn echter test die geen value hebben zoals ene login test. Deze test is gelukt of niet. Deze informatie zit dus in de status. Vandaar dat er met 2 velden gewerkt wordt. Men zou deze velden evenwel kunnen samensteken door bv negatieve waarden te gebruiken voor een foute status. Al zorgt dit voor overbodige complexiteit. Verder is het moeilijker om consequent te blijven. Als de status volledig op server-side bepaald wordt, is deze overal dezelfde bij dezelfde waarde.

4.6 Besluit

De laatste uitwerking lijkt mij de meest logische. Het is zeer intuïtief om de ping van de virtualwall op te halen met het commando: test/ping/value/virtualWall/ of als we de eerste parameter weglaten ping/value/virtualWall.

5 Conclusie

Er zijn 3 uitwerkingen. De eerste heeft per test een aparte functie. De tweede is een hybride uitwerking waarbij het opvragen van testresultaten al gegroepeerd wordt. De derde is volledig generisch. De derde lijkt mij het eenvoudiste omdat gebruikers dan gene onderscheid van getTest, getTestbed en getResult moeten onthouden. Verder kan er dan eenvoudig gelinkt worden ping/frequency geeft dan de frequentie weer terwijl ping/value de waardes bevat. Op deze manier kunnen we onze webservice beschouwen als een verzameling objecten waarvan we waarden opvragen. De eerste parameter objectType kan behouden worden om te vermijden dat de webservice onoverzichtelijk wordt door de generische opbouw.