Automated testbed Monitoring for jFed

Andreas De Lille

March 12, 2014

1 Inhoud

Dit document gaat dieper in op het ontwerp van de webservice. Zo zullen de verschillende mogelijke calls uitgelegd worden. Ook de interne werking van de webservice en de opbouw van de achterliggende database komen aan bod. Het is de bedoeling dat ook het actualiseren van de monitoring informatie via deze service verloopt.

2 Api Calls

We kunnen de calls op delen in 2 grote groepen.

- Results : Het opvragen van resultaten van een test.
- Configuration : Het opvragen, aanpassen en aanmaken van testen en de bijhorende configuratie.

3 Calls - Results

Voor het opvragen van resultaten wordt gebruik gemaakt van een generische functie.

3.1 Functies

Er zijn een aantal functies voorzien voor het opvragen van de resultaten.

- last : Deze functie geeft het laatste/ de laatste resultaat/resultaten terug (per test,testbed).
- list : Deze functie geeft een lijst resultaten die voldoen aan de opgegeven parameters. Zo kan deze lijst bijvoorbeeld beperkt worden tot een resultaten die overeenkomen met een bepaalde status of vallen tussen bepaalde data.
- 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.

3.2 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. Ook wanneer er enkel een till opgegeven is, worden de laatste resultaten voor de opgegeven tijd teruggegeven. Wanneer er enkel een from parameter opgegeven is, dan worden de eerste x resultaten startend vanaf de from terug gegeven.
- from : Geeft aan vanaf welke timestamp er gezocht moet worden.
- till : Geeft aan tot welke timestamp de resultaten gezocht moeten worden.
- status : Geeft aan welke status de resultaten moeten hebben, bijvoorbeeld Fail,Warn of Succes.
- resultid : Geeft het id van een testresultaat waarvan men een detail weergave wil.
- testname : De naam van de test. Dit is ENKEL nodig bij stitching tests, omdat deze over meerdere testbeds gaan.

Indien bij tests of bij testbeds de waarde all wordt gegeven, worden respectievelijk alle tests of alle testbeds teruggegeven. Ook opgeven van lijsten is mogelijk.

3.3 Voorbeelden

Hieronder zal ik een aantal voorbeelden van calls en bijhorende antwoorden geven. Voorlopig zijn deze voorbeelden enkel beperkt tot het opvragen van data met behulp van HTTP GET requests.

3.3.1 last?testbed=urn-testbed0&test=ping&count=2

Deze call geeft voor testbed0 de laatste 2 ping resultaten terug.


```
[{
    "testname": "ping",
    "testbedId": "urn-testbed0",
    "planId": 103,
    "resultId": 1499,
    "log": "http://www.urn-testbed0com/Logs/ping/log1499",
    "results": [{
        "name": "pingValue",
        "value": 39
      }],
    "timestamp": "2014-03-14 18:08:38"
  },{
    "testname": "ping",
    "testbedId": "urn-testbed0",
    "planId": 103,
    "resultId": 1500,
    "log": "http://www.urn-testbed0com/Logs/ping/log1500",
    "results": [{
        "name": "pingValue",
        "value": 35
      }],
    "timestamp": "2014-03-14 18:08:39"
}]
```

3.3.2 last?testbed=ALL&test=ping

Deze call geeft voor elk testbed de laatste ping resultaten terug.


```
[{
    "testname": "ping",
    "testbedId": "urn-testbed0",
    "planId": 103,
    "resultId": 1452,
    "log": "http://www.urn-testbed0com/Logs/ping/log1452",
    "results": [
      {
        "name": "pingValue",
        "value": 35
      }
    ],
    "timestamp": "2014-03-14 18:03:37"
  },{
    "testname": "ping",
    "testbedId": "urn-testbed1",
    "planId": 104,
    "resultId": 1453,
    "log": "http://www.urn-testbed1com/Logs/ping/log1453",
    "results": [
      {
        "name": "pingValue",
        "value": 56
      }],
    "timestamp": "2014-03-14 18:03:38"
  },{
    "testname": "ping",
    "testbedId": "urn-testbed2",
    "planId": 105,
    "resultId": 1454,
    "log": "http://www.urn-testbed2com/Logs/ping/log1454",
    "results": [{
        "name": "pingValue",
        "value": 39
      }],
    "timestamp": "2014-03-14 18:03:39"
}]
```

3.3.3 last?test=stitching&testname=stitching-name2

Deze functie geeft een detail resultaat terug van de stitchingtest met naam 'stitching-name2'. Er kan eventueel een globaal resultaat toegevoegd worden om weer te geven of de stitching test in zijn geheel is gelukt.


```
{
  "testname": "stitching-name2",
  "testbeds": [
    "urn-testbed0",
    "urn-testbed1",
    "urn-testbed2"],
  "planId": 80,
  "resultId": 2360,
  "log": "http://www.stitching-name2com/Logs/stitching-name2/log2360",
  "results": [{
      "name": "setup",
      "value": "succes"
    },{
      "name": "getUserCredential",
      "value": "succes"
    },{
      "name": "generateRspec",
      "value": "succes"
    },{
      "name": "createSlice",
      "value": "succes"
    },{
      "name": "initStitching",
      "value": "succes"
    },{
      "name": "callSCS",
      "value": "succes"
    },{
      "name": "callCreateSlivers",
      "value": "succes"
    },{
      "name": "waitForAllReady",
      "value": "succes"
    },{
      "name": "loginAndPing",
      "value": "succes"
    },{
      "name": "callDeletes",
      "value": "succes"
    }],
  "timestamp": "2014-03-14 18:03:39"
}
```

3.3.4 Average?testbed=urn-testbed0&test=ping &from=2014-03-1T18:03:37 &till=2014-03-11T18:03:37

Deze call geeft het gemiddelde van alle waarden van de ping test op testbed0 terug.


```
{
  "testname": "ping-average",
  "testbedId": "urn-testbed0",
  "results": [
    {
      "name": "average-pingValue",
      "value": 38.666666666667
    }
  ],
  "from": "2014-03-1 18:03:37",
  "till": "2014-03-11 18:03:37"
}
```


3.3.5 list?testbed=urn-testbed0&test=ping &till="2014-03-14T18:08:39 &count=2

Deze call geeft een lijst van de 2 laatste pingtests voor 2014-03-14 18:08:39 op testbed0 terug.


```
[{
    "testname": "ping",
    "testbedId": "urn-testbed0",
    "planId": 103,
    "resultId": 1499,
    "log": "http://www.urn-testbed0com/Logs/ping/log1499",
    "results": [{
        "name": "pingValue",
        "value": 39
      }],
    "timestamp": "2014-03-14 18:08:38"
  },{
    "testname": "ping",
    "testbedId": "urn-testbed0",
    "planId": 103,
    "resultId": 1500,
    "log": "http://www.urn-testbed0com/Logs/ping/log1500",
    "results": [{
        "name": "pingValue",
        "value": 35
      }],
    "timestamp": "2014-03-14 18:08:39"
}]
```

4 Calls - configuration

Deze calls hebben het doel om de configuratie van een test op te vragen.

4.1 Functies

Er zijn (uiteraard) ook een aantal functies voor het opvragen van de testconfiguratie.

- TestDefinition : Geeft de definitie van een testweer. Deze bevat informatie over de naam, het commando. Verder duidt deze definitie ook aan welke parameters er nodig zijn en welke waarden er terug gegeven worden. Deze waarden zijn echter niet ingevuld. Het is de bedoel om hier bijvoorbeeld te definiëren dat een pingtest bijvoorbeeld bestaat uit een ping commando en dat deze een timeout en een testbed moet meekrijgen. Samengevat kan gesteld worden dat dit de beschrijving van een type test is.
- TestInstance : Hierbij wordt een instance teruggegeven. Dit is de concrete invulling van een testDescription. Daarbij wordt de test gecombineerd met de ingevulde parameters. Zo zal hier ingevuld worden dat er op testbed2 om de 5 minuten een pingtest moet uitgevoerd worden.

Deze opsplitsing is noodzakelijk om flexibiliteit aan te bieden. Een ping test wordt op meerdere testbeds uitgevoerd. Door deze opbouw moet een ping test maar eenmalig gedefiniëerd worden. Als we de vergelijking met mvc maken, geeft de interface van een klasse waar, de instance kan gezien worden als een object van die klasse.

4.2 Parameters

- test : Deze parameter duidt aan welke test opgevraagd wordt.
- instance : Deze parameter duit aan welk instance opgevraagd wordt.

Het is ook mogelijk om als waarde ALL of een lijst van id's op te geven.

4.3 Voorbeelden

Hieronder worden een aantal mogelijke calls weergegeven. Voorlopig zijn deze voorbeelden beperkt tot het opvragen van data via http get requests.

4.3.1 testDefinition?test=stitching

Deze call geeft de beschrijving van een stitching test terug.


```
{ "definitionId": "stitching",
  "command": "stitch",
  "parameters": [{
      "name": "topology",
      "type": "string"
    },{
      "name": "testbeds",
      "type": "testbed[]"}],
  "return": [{
      "name": "setup",
      "type": "string",
      "description": "succes?"
    },{
      "name": "getUserCredential",
      "type": "string",
      "description": "succes?"
    },{
      "name": "generateRspec",
      "type": "string",
      "description": "succes?"
    },{
      "name": "createSlice",
      "type": "string",
      "description": "succes?"
    },{
      "name": "initStitching",
      "type": "string",
      "description": "succes?"
    },{
      "name": "callSCS",
      "type": "string",
      "description": "succes?"
    },{
      "name": "callCreateSlivers",
      "type": "string",
      "description": "succes?"
    },{
      "name": "waitForAllReady", "type": "string", "description": "succes?"
    },{
      "name": "loginAndPing", "type": "string", "description": "succes?"
    },{
      "name": "callDeletes", "type": "string","description": "succes?"
}]}
```

4.3.2 testDescription?test=all

Hierbij worden alle testbeschrijvingen teruggegeven.


```
[{
    "definitionId": "ping",
    "command": "ping",
    "parameters": [{
        "name": "timeout", "type": "int"
      },{
        "name": "testbed", "type": "string"
    }],
    "return": [{
        "name": "pingValue", "type": "int", "description": "pingValue"
     }]
    },{
    "definitionId": "stitching",
    "command": "stitch",
    "parameters": [{
        "name": "topology", "type": "string"
      },{
        "name": "testbeds","type": "testbed[]"
     }],
    "return":
    [{
        "name": "setup", "type": "string",
        "description": "succes?"
      },{
        "name": "getUserCredential", "type": "string", "description": "succes?"
      },{
        "name": "generateRspec", "type": "string", "description": "succes?"
      },{
        "name": "createSlice", "type": "string", "description": "succes?"
      },{
        "name": "initStitching","type": "string", "description": "succes?"
      },{
        "name": "callSCS","type": "string","description": "succes?"
      },{
        "name": "callCreateSlivers","type": "string","description": "succes?"
      },{
        "name": "waitForAllReady", "type": "string", "description": "succes?"
      },{
        "name": "loginAndPing", "type": "string", "description": "succes?"
      },{
        "name": "callDeletes", "type": "string", "description": "succes?"
}]}]
```

4.3.3 testInstance?Instance=all

Deze call geeft alle geplande tests terug.


```
[{
    "definitionId": "ping",
    "parameters": [{
        "name": "timeout", "value": 106
       },{
        "name": "testbed","value": "urn-testbed0"
    }],
    "frequency": 1488,
    "planId": 0
  },{
    "definitionId": "ping",
    "parameters": [{
        "name": "timeout","value": 136
      },{
        "name": "testbed","value": "urn-testbed1"
    }],
    "frequency": 575,
    "planId": 1
  },{
    "definitionId": "ping",
    "parameters": [{
        "name": "timeout","value": 106
      },{
        "name": "testbed","value": "urn-testbed2"
    }],
    "frequency": 1637,
    "planId": 2
  },{
    "definitionId": "stitching",
    "parameters": [{
        "name": "topology","value": "ring"
      },{
        "name": "testbeds",
        "value": [
          "urn-testbed0","urn-testbed3"
    ]}],
    "frequency": 9362,
    "planId": 3
  },
```

```
 {
    "definitionId": "stitching",
    "parameters": [{
        "name": "topology","value": "ring"
      },{
        "name": "testbeds",
        "value": [
          "urn-testbed0","urn-testbed3"
    ]}],
    "frequency": 6991,
    "planId": 4
  },{
    "definitionId": "stitching",
    "parameters": [{
        "name": "topology","value": "ring"
      },{
        "name": "testbeds",
        "value": [
          "urn-testbed0","urn-testbed3"
    ]}],
    "frequency": 8458,
    "planId": 5
}]
```


4.3.4 testInstance?Plan=2

Deze call geeft de instance met id 2 terug.


```
{
  "definitionId": "ping",
  "parameters": [
    {
      "name": "timeout",
      "value": 108
    },
    {
      "name": "testbed",
      "value": "urn-testbed2"
    }
  ],
  "frequency": 3046,
  "planId": 2
}
```

5 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.

6 Database

Hieronder wordt het ontwerp van de database uitgelegd. In een eerste versie wordt het aantal tabellen nog beperkt om de uitwerking makkelijker en sneller te maken. Later zullen 'hardcoded' lijsten met behulp van extra tabellen in de database zitten. De eerste versie zal ook 2 tests bevatten de ping test met attributen testbed en timeout. De tweede test is de stitching test die een lijst van testbeds meekrijgt als parameter, samen met een opstelling zoals ring of line.

6.1 Tabellen

De database zal een postgresql database zijn en zal bestaan uit volgende tabellen.

- testbeds : Deze tabel zal informatie over de testbeds bijhouden
- testDefinition : Deze tabel zal de testbeschrijving/testdefinitie bijhouden. Zo wordt hier bijvoorbeeld gedefiniëerd dat een ping test bestaat uit een ping commando. Ook de parameters van een pingtest worden hier aangegeven. Zo zal een pingtest bijvoorbeeld een testbed moeten meekrijgen waar de pingtest op uitgevoerd moet worden en een timeout. Ook zal er gedefinieerd zijn dat deze test een waarde teruggeeft. Later kan er eventueel gedefinieerd worden wat er moet gebeuren met die waarde (bijvoorbeeld mailen indien een testbed offline blijkt te zijn). Deze parameters zullen in de eerste versie nog 'hardcoded' in de tabel zitten (bv een array in json formaat). Later kan door gebruik te maken van een tabel dit gedeelte eventueel 'opgekuist' worden.
- testInstance : Hierbij worden de instanties van de testdefinities bijgehouden. Deze tabel zorgt voor de link tussen de testdefinitie en de ingevulde parameters. In een eerste versie zitten deze parameters echter 'hardcoded' als een hash in json formaat opgeslagen. Later kan een extra tabel dit 'opkuisen'.
- Results : Deze tabel zal de resultaten opslaan van de tests. Merk op dat indien de resultaten multivalued zijn (zoals de stitching test) er een array wordt opgeslagen. In een eerste versie zal dit zijn door de json van een array op te slaan.

Hieronder zal ik kort de (eerste) versie van de tabellen bespreken. De precieze aantal kolommen is momenteel nog niet vastgelegd. De nadruk bij de eerste versie ligt dan ook op de functionaliteit. De eerste versie moet aantonen dat het gekozen ontwerp flexibel genoeg is om aan de eisen te voldoen. Merk ook de opsplitsing testDefinition ¡=¿ testInstance.

6.1.1 Testbeds

Deze tabel houdt de verschillende testbeds bij en bevat volgende kolommen :

- testbedId : Het id van het testbed. Aangezien elk testbed een uniek urn heeft zal dit waarschijnlijk gebruikt worden als testbedid.
- name : de naam van het testbed.

6.1.2 TestDefinition

Deze tabel houdt de definities van de tests bij. Hierbij wordt een test abstract omschreven als een commando met een aantal parameters en een of meerdere return waarden. De tabel bevat volgende kolommen :

- testName : Dit is de naam van een test en moet tevens uniek zijn.
- commando : Het commando (of script) dat uitgevoerd moet worden.
- parameters : Een lijst van parameters die een commando nodig heeft. In de eerste versie is deze lijst opgeslagen als de json encodering van een array. Voor elke parameter is er een naam , een type en een beschrijving.
- return : Met return wordt een lijst weergegeven van de return waarden van een test. Bij een ping is dit simpelweg een ping waarde. Bij een stitching test is dit echter een lijst van resultaten van subtests. Ook dit wordt in een eerste versie eenvoudig opgeslagen. Voor elke waarde wordt een naam , een type en een beschrijving bijgehouden.

6.1.3 testInstance

Deze tabel zal later de testdefinition koppelen aan de tabel met parameters. In de eerste versie zal hij de waarden hardgecodeerd bevatten. In deze tabel wordt er bijgehouden der er om de 5 minuten een ping test uitgevoerd moet worden op een testbedX met een timeout Y. De tabel bevat volgende kolommen :

- testInstanceId : Een id om de testinstance mee aan te duiden.
- TestDefinition : Deze geeft weer om welk type test het gaat.
- Frequency : De frequentie waarmee de test uitgevoerd moet worden.
- Parameters : Een (in json gecodeerde) lijst van parameters.

6.1.4 Results

Deze tabel zal de resultaten bijhouden van de test. Hij bevat volgende kolommen.

- testInstance : Het 'type' van de test
- resultid : Later gebruikt om detail weergave op te vragen van een resultaat.
- results : een (in json gecodeerde) lijst met resultaten.
- log : De link naar de link van de log file. Deze zal beschikbaar staan op het testbed, via de url is het mogelijk om hem op te vragen.
- timestamp : Een tijdsaanduiding.

6.2 Uitwerking - volgende stappen

De uitwerking van deze webservice zal eerst bestaan uit het opvragen van testresultaten en testconfiguratie. Na de aanpassingen in de database(uitwerken hoe de lijsten opgeslagen worden), zal het ook mogelijk moeten zijn om testen aan te passen en te wijzigingen. Verder moet het ook mogelijk zijn om de nieuwe resultaten op te slaan in de databank.