Overview about the feature for connection to the perfSONARUIApp of the perfsonarServer

Summary

The package perfsonarserver.appConnect contains the functionality to receive requests by the perfSONARUIApp (referred to as app in the following text), handle these requests and sending a response to the app. The requests are made via HTTP POST, the POST body contains the request as JSON data. The JSON data is parsed, passed to the database and the returned database response is transformed back into JSON data, which will then be sent to the app as HTTP POST response. The connection feature is able to handle multiple requests at a time via multithreading. For details see the class information below.

Class overwiev

RequestTO

Implementation of a transfer object (referred to as TO later on), which is filled with the data received in a request and then passed on to the database
Listener: Server thread, waits for incoming connections. Starts a worker thread and connects incoming connection to it.

Communicator

Worker thread started by Listener. Here the POST request is processed by storing the request JSON data in a file and then calling the appropriate methods to parse the JSON data to get a TO with which to perform the database request. Then the JSON data for the response is created (see JsonHandler) and the response is sent to the app.

JsonHandler

Uses Ralf Sternberg's minimal-json parser (https://github.com/ralfstx/minimal-json) to parse the JSON request data and to create the JSON data for the response.

DatabaseRequest

Uses the RequestTO to perform a database request. For each request type the according database request is performed. The returned data is stored in a JSON array and returned to the caller – the Communicator worker thread.

For detailed functionality see JavaDoc and inline comments in the code itself.

JSON requests and responses

Requests

These are exemplary JSON requests for requesting delay jitter loss-services, source interfaces, destination interfaces for a source interface and delay jitter loss data for an interface pair respectively. The structure of the JSON requests for the other measurements is according to these, for the request type (FeatureName in the JSON data) see the switch/case arguments in DatabaseRequest.

```
{
      "FeatureName": "DelayJitterLossGetService",
      "Service": null,
      "SourceInterface": null,
      "DestinationInterface": null,
      "StartTime": null,
      "EndTime": null,
      "GetNonCachedData": true
}
```

```
{
      "FeatureName": "DelayJitterLossGetSourceInterfaces",
      "Service": "X-WiN",
      "SourceInterface": null,
      "DestinationInterface": null,
      "StartTime": null,
      "EndTime": null,
      "GetNonCachedData": true
}
```

```
{
      "FeatureName": "DelayJitterLossGetDestinationInterfaces",
      "Service": "X-WiN",
      "SourceInterface": "Bremerhaven_AWI",
      "DestinationInterface": null,
      "StartTime": null,
      "EndTime": null,
      "GetNonCachedData": true
}
```

```
{
      "FeatureName": "DelayJitterLossGetData",
      "Service": "X-WiN",
      "SourceInterface": "Augsburg_DFN",
      "DestinationInterface": "Aachen_DFN",
      "StartTime": "2013-03-31 23-00-00-000",
      "EndTime": "2013-03-31 23-59-00-000",
      "GetNonCachedData": true
}
```

Responses

These are the responses to the exemplary requests. The responses for the interface requests and the data request are shortened for clarity. The responses are created without tabs or line breaks, these are added for clarity

```
{
      "service":[
            {"Service":"GEANT"},
            {"Service":"LHCOPN"},
            {"Service":"X-WiN"}
      ]
}
```

```
{
      "Info":{
            "Service":"X-WiN"
      },
      "Interfaces":[
            {"SourceInterface":"Bremerhaven_AWI"},
            {"SourceInterface":"Bremen_DFN"},
            {"SourceInterface":"Saarbruecken_Uni"},
            …,
            {"SourceInterface":"Greifswald_DFN"}
      ]
}
```

```
{
      "Info":{
            "Service":"X-WiN",
            "SourceInterface":"Bremerhaven_AWI"
      },
      "Interfaces":[
            {"DestinationInterface":"Hannover_Uni"},
            {"DestinationInterface":"Bayreuth_Uni"},
            {"DestinationInterface":"Berlin_TUB_DFN"},
            …,
            {"DestinationInterface":"Kassel_DFN"}
      ]
}
```

```
{
      "Info":{
            "Service":"X-WiN",
            "SourceInterface":"Augsburg_DFN",
            "DestinationInterface":"Aachen_DFN",
            "StartTime":"2013-03-31 23-00-00-000",
            "EndTime":"2013-03-31 23-59-00-000",
            "GetNonCachedData":true
      },
      "MeasuredValues":[
            {"date":"2013-03-31 23-00-15-961",
            "maxDelay":0.00615596771240234,
            "minDelay":0.00614500045776367,
            "maxJitter":8.82148742675781E-6,
            "minJitter":-7.86781311035156E-6,
            "loss":0},
            {"date":"2013-03-31 23-01-15-960",
            "maxDelay":0.00615596771240234,
            "minDelay":0.00614500045776367,
            "maxJitter":9.77516174316406E-6,
            "minJitter":-7.86781311035156E-6,
            "loss":0},
            …,
            {"date":"2013-03-31 23-58-15-961",
            "maxDelay":0.0061490535736084,
            "minDelay":0.00613903999328613,
            "maxJitter":1.00135803222656E-5,
            "minJitter":-9.05990600585938E-6,
            "loss":0}
      ]
}
```

To reduce the amount of data that has to be sent to the app, the requests for source and destination interfaces are seperated, even though the measurement archive servers send these as pairs. First the app requests the source interfaces, then after selecting one of the source interfaces the according destination interfaces are requested by the app.

To be done:

- The database requests are only fully implemented for the delay jitter loss feature. The other features are prepared rudimentary, but full functionality was not implemented tue to the database not being ready.
- Exception handling is minimal