

CenturyLink IT Services

Web Service Utility Configuration

Abstract

This document represents the need and setup information of service utilty.

# Introduction

Service Profiler is a product developed by internally where we can track the request, response and total time taken by each service. From service profiler we can also track the total time taken of sub services inside each service.

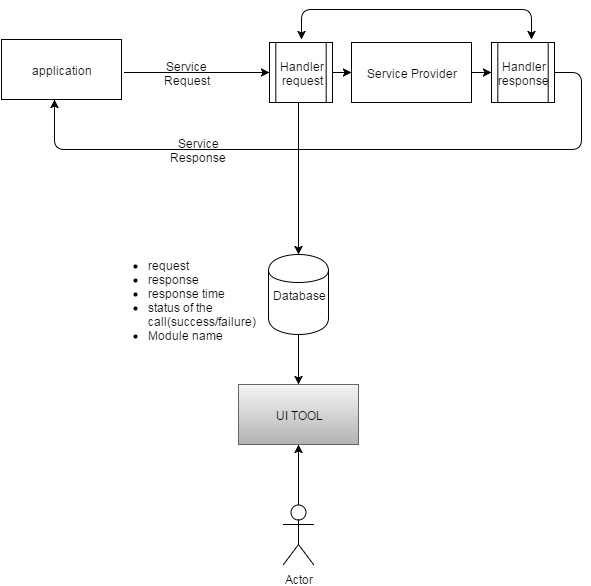
# 2.0 How it is useful

From service profiler we can track the records of each request and response with respective time in database. Consider a scenario there is an issue with data in the production and we want to debug it. Since each records are stored in database to respective time we can search for specific interval of time in table to know the exact request and response of that data instead of debugging in the logs.

# 3.0 Advantages

* Helps to improve the performance of application
* Track the records of request, response and total time taken by each service.
* Easy to integrate with application
* Pentaho dashboards to view the records
* Multiple dashboards for multiple functionality
* Time saver to debug the request and response.

# 4.0 High Level Design



# 5.0 Database design

There are two tables to store the service calls details.

1. Web\_service\_utility: This table mainly stores the parent service call and other details.

2. Child service: It stores all the child service call and response time.

Untitled Diagram3

**CREATE** **TABLE** `web\_service\_utility` (

`sl\_no` **INT**(11) **NOT** **NULL** **AUTO\_INCREMENT**,

`request\_date` **DATETIME** **NOT** **NULL** **DEFAULT** **CURRENT\_TIMESTAMP**,

`module` **VARCHAR**(100) **NULL** **DEFAULT** **NULL**,

`request\_xml` **VARCHAR**(4000) **NULL** **DEFAULT** **NULL**,

`response\_xml` **VARCHAR**(4000) **NULL** **DEFAULT** **NULL**,

`total\_elapsed\_time` **MEDIUMTEXT** **NULL**,

`call\_status` **VARCHAR**(20) **NULL** **DEFAULT** **NULL**,

**PRIMARY** **KEY** (`sl\_no`)

);

Untitled Diagram1

**CREATE** **TABLE** `child\_service` (

`cs\_id` **INT**(11) **NOT** **NULL** **AUTO\_INCREMENT**,

`moduleName` **VARCHAR**(100) **NULL** **DEFAULT** **NULL**,

`time` **MEDIUMTEXT** **NULL**,

`utility\_id` **INT**(11) **NULL** **DEFAULT** **NULL**,

**PRIMARY** **KEY** (`cs\_id`),

**INDEX** `utility\_id` (`utility\_id`),

**CONSTRAINT** `child\_service\_ibfk\_1` **FOREIGN** **KEY** (‘cs\_id`) **REFERENCES** `web\_service\_utility` (‘sl\_no`)

)

# 6.0 Pentaho Setup

First we need to set the PENTAHO\_JAVA\_HOME.Suppose java is installed in “C:\Program Files\Java\jdk1.7.0\_04\bin”.Please follow below steps:

1. Click Start.
2. Right-click on Computer, click Properties.
3. In the left pane, click Advanced system settings.
4. Select the Advanced tab, then click Environment Variables…
5. In System Variables, scroll till you find Path.
6. Select Path, click Edit…
7. Add the Java bin folder to  your path by appending ;C:\Program Files\Java\jdk1.7.0\_04\bin to the end of the Path Variable Value.
8. Click OK.
9. Click New…
10. In Variable name, enter: PENTAHO\_JAVA\_HOME
11. In Variable value, enter: C:\Program Files\Java\jdk1.7.0\_04
12. Click OK.
13. Click OK.
14. Click OK.

15. Open Powershell.

16. Then type javac -version.

17. For each command above, you should see the correct Java version  (for me its jdk1.7.0\_04)

18. Open a command prompt.

19. Type echo %PENTAHO\_JAVA\_HOME%

20. You should see the following as output: C:\Program Files\Java\jdk1.7.0\_04

# 7.0 Installation of Pentaho BI Server

* Now that we have Java installed we can get on with our main task of installing the Pentaho BI Server.
* Download the Pentaho BI Server from <http://wiki.pentaho.com/display/COM/Latest+Stable+Builds>. I’m using the current stable build which is **biserver-ce-4.8.0-stable.zip**
* Open Windows Explorer and create new folder: **C:\Program Files\Pentaho**.
* Unzip **biserver-ce-4.8.0-stable.zip**
* Next, copy the **biserver-ce** and **administration-console** folder into

**C:\Program Files\Pentaho**.

To start the pentaho server open Windows Explorer to C:\Program Files\Pentaho\biserver-ce and double-click  On “start-pentaho.bat” and open a browser to <http://localhost:8080/pentaho/Login>

and the pentaho starts.

(default login:UserName:joe and Password:password)

Note:If you are not able to start the pentaho with above steps please edit the start-pentaho.bat

And replace the

set CATALINA\_OPTS=-Xms256m -Xmx768m -XX:MaxPermSize=512m -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000

To start the Pentaho Administration console Open Windows Explorer to **C:\Program Files\Pentaho\administration-console** and double-click on “start-pac.bat”.

Open a web browser to  [http://localhost:8099](http://localhost:8099/)

Enter a **User Name** of **admin**.

Enter a **Password** of **password**.

Click **Log In**.

# 8.0 Database setup

Step 1: Download MySQL

Download MySQL from dev.mysql.com/downloads/. Follow MySQL Community Server, Windows and download the “Without installer” version.

Step 2: Extract the files

We will install MySQL to C:mysql, so extract the ZIP to your C: drive and rename the folder from “mysql-x.x.xx-win32” to “mysql”.

Step 3: Test your installation

The MySQL server is started by running C:/mysql/bin/mysqld.exe. Open a command box (Start > Run > cmd) and enter the following commands:

1. cd C:/mysql/bin/
2. mysqld

This will start the MySQL server which listens for requests on localhost port 3306. You can now start the MySQL command line tool and connect to the database.

Step 4: Open another command box and enter:

1. cd C:/mysql/bin/
2. mysql -u root

This will show a welcome message and the mysql> prompt. Enter “show databases;” to view a list of the pre-defined databases.

# Integration of Service Profiler With Application

Configuration:

Step 1: Service Profiler jar need to be copied in lib folder.

Step 2: Configuring Handler

handler.xml(name configurable but same need to be used in Soap service class) need to be created in any one of the packages inside src/main/Java source folder.

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<javaee:handler-chains xmlns:javaee="http://java.sun.com/xml/ns/javaee"

xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<javaee:handler-chain>

<javaee:handler>

<javaee:handler-class>com.service.profiler.handler.WebServiceStatsHandler</javaee:handler-class>

</javaee:handler>

</javaee:handler-chain>

</javaee:handler-chains>

Step 3: Annotate handler

At the top of Soap service class we need to annotate the handler.xml

@HandlerChain(file="handlers.xml")

Step 4: Capture child services

If we want to capture the time taken by each sub services then we need to configure the below steps.

Long startTime1 = ServiceHelper.startTimer();

// service1 call

ServiceHelper.stopTimer(startTime1, "name of sub service");

Long startTime2 = ServiceHelper.startTimer();

// service2 call

ServiceHelper.stopTimer(startTime2, "service 2");

WebServiceStatsHandler.updateServices(ServiceHelper.serviceMap);

Step 5: Configuring DB

To configure the db. Create dbConfig.properties and place it in src/main/resources(classpath) source folder.