

## SDN Community Contribution

(This is not an official SAP document.)

### Disclaimer & Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.

## Applies To:

Java Stack of SAP NetWeaver (Web) Application Server SP12 (probably also lower SPs)

## Summary

This article discusses how to trace the performance of Java applications on SAP Web AS without using cost-intensive profilers and using a decompiler to inspect nearly every class deployed on your Web AS.

**By:** Helmut Tammen

**Company:** HT Systemlösungen, [www.htsystemloesungen.de](http://www.htsystemloesungen.de)

**Date:** Dec. 4, 2005

## Table of Contents

Applies To:.....	2
Summary .....	2
Table of Contents .....	2
Introduction.....	3
Preconditions.....	3
<i>Performance Trace service</i> .....	3
Starting Application tracing.....	3
Running the Application.....	4
Analyze the Result.....	4
Decompile the programs .....	5
Decompiler Configuration .....	5
View Decompiled Code .....	6
View Decompiled Code from Classes .....	7
Conclusion.....	7
Author Bio.....	7

## Introduction

This How-to guide describes the basic usage of the Performance Trace service in the J2EE Engine of SAP Web AS and how you can decompile nearly each class deployed at your Web AS.

If you sometimes have performance leaks in your applications (or in applications written by others) you can of course use a professional profiler to analyse your program. But the J2EE engine of Web AS already gives you the possibility to measure the performance of applications without having too much effort.

A nice side effect of the performance tracer is that you can simply decompile any tested application.

## Preconditions

There are only a few preconditions that have to be fulfilled.

- You need an administration account at the Web AS.
- If you want to decompile the programs you need to have an external decompiler installed at your local PC.

## Performance Trace service

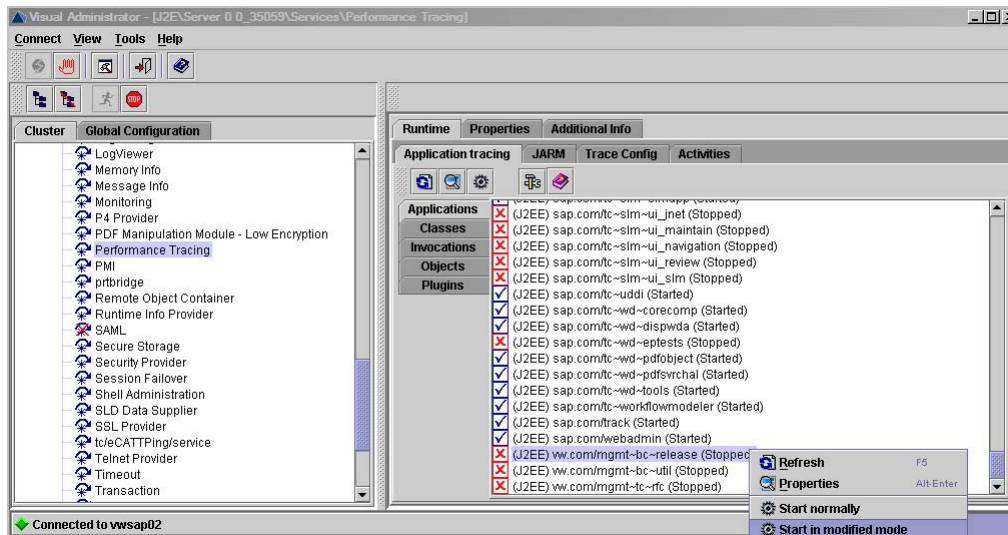
The Performance Trace service can be accessed via the Visual Administrator. You find it in the tree at the left side under Server ... --> Services --> Performance Tracing.

The service offers different performance tracing options. In this document we will just have a look at application tracing.

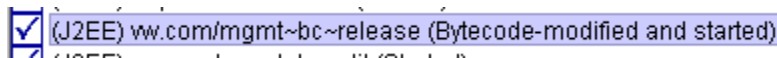
## Starting Application tracing

Under the tabs Runtime --> Application tracing --> Applications you see a list of the applications that are deployed at your server. There you look for the application you want to do performance tracing for. In the context menu (right mouse click) of the item you choose the menu item (Re)Start in modified mode.

Now the application will be restarted with modified byte code so that performance tracing can be done.



The text of the restarted application will change after a while.



## Running the Application

After you have started the application with modified bytecode you can run it. Keep in mind that the application is much slower than normally. This is obviously because of the tracing.

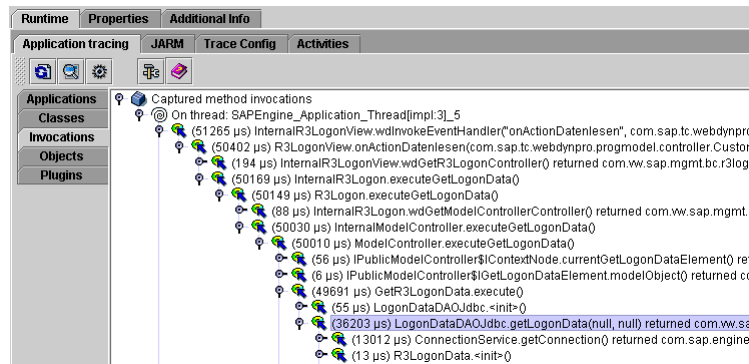
When ready go back to Visual Admin to have a look at the measurement results.

## Analyze the Result

To analyse the result of your performance test go to the Performance tracing service and select the item 'Invocations' under the tabs 'Runtime' --> 'Application tracing'.

You will see one or more threads depending on how many actions you performed in your application.

Select the thread you are interested in and navigate through the tree of called methods until you've found the method(s) that make your application slow (take a long time to process). In the front of each method that has been called you see the time in microseconds that it took to perform.



It must not be said that you should now have a look at your code to do some optimisation.

## Decompile the programs

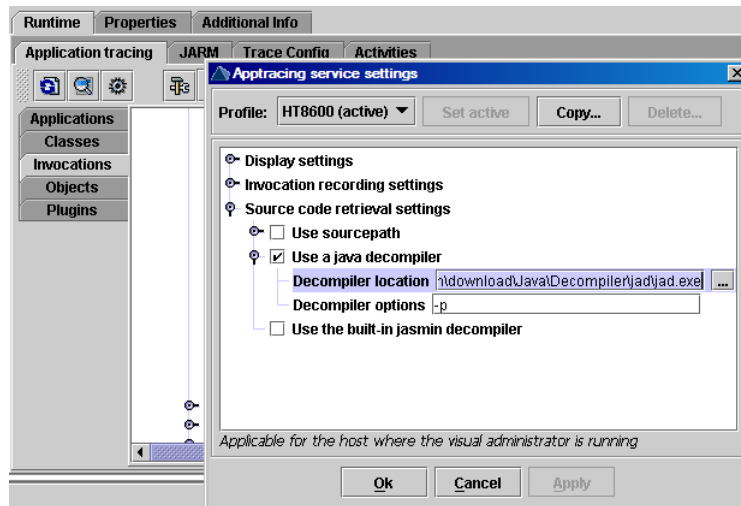
If the program you analysed is not written by yourself you can use a decompiler to have a look at the source code.

### Decompiler Configuration

First you have to configure the Performance tracing service to use a locally installed decompiler. In my example I used the JAD decompiler (<http://www.kpdus.com/jad.html>). Please keep in mind that it can only be used free of charge for none commercial use.

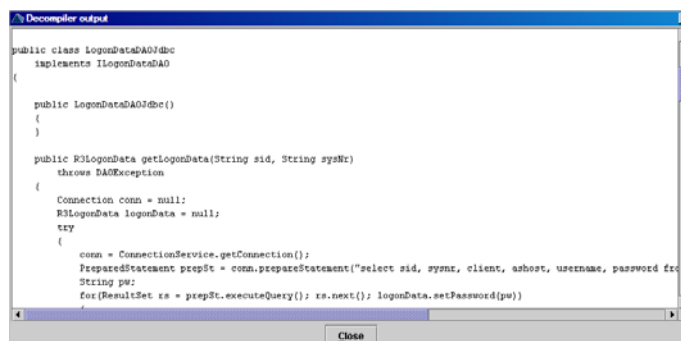
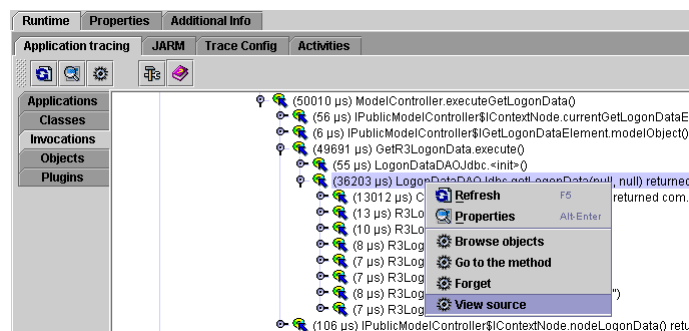
Under the tabs Runtime --> Application tracing click at the tool icon. The settings dialog will open. Here you open the tree "Source code retrieval settings" and select the checkbox in front of "Use a java decompiler." In the input field "Decompiler location" you fill in the name of your decompiler application (e.g. .../.../jad.exe). Alternatively you can browse for it. The decompiler option depends on your selected decompiler. For JAD the "-p" option means that the output of the decompiler is written to STDOUT which has to be the case if you want to view the decompiled code directly in Visual Admin.

Save your settings by clicking OK.



## View Decompiled Code

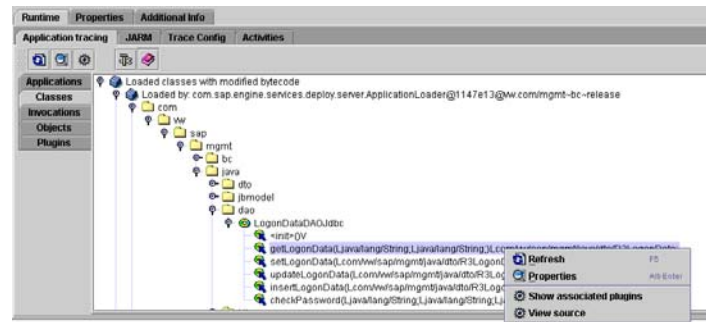
After you configured the Performance tracing service go back to the Invocations menu item and open the context menu of the method you are interested in. Here select View source.



You will then see the decompiled code in a dialog box.

## View Decompiled Code from Classes

Another and may be more convenient way to have a look at the sourcecode is to change to the classes view. Here you find a list of all bytecode modified classes in a hierarchical view. In the context menu of a class or method you will find the menu item View source again.



## Conclusion

With the Performance tracing service SAP offers a service that is sufficient for identifying performance leaks in your applications for most cases. It will of course not substitute a professional profiler that is recommended for larger or time critical projects but you can do basic performance tests with just a view clicks out of the box and without investing in further tools.

By the way it is not only possible to decompile your own code. You can also have a look at how SAP developed their programs. This sometimes might be very useful.

[Editor's Note: See [Performance Trace Overview](#) in SAP Help for additional information.]

## Author Bio



Helmut Tammen has been an independent SAP NetWeaver consultant for more than five years. In this time he started with development and consulting in R/3 systems using ABAP. In the last 3 ½ years he focused on Enterprise Portal, J2EE, Web Dynpro, NWDI. Before his time with SAP products he designed and developed applications under MS-DOS (yes, that was possible) and Windows using mostly OO-technology and languages like Clipper, C, C++, Delphi and Java since 1990.