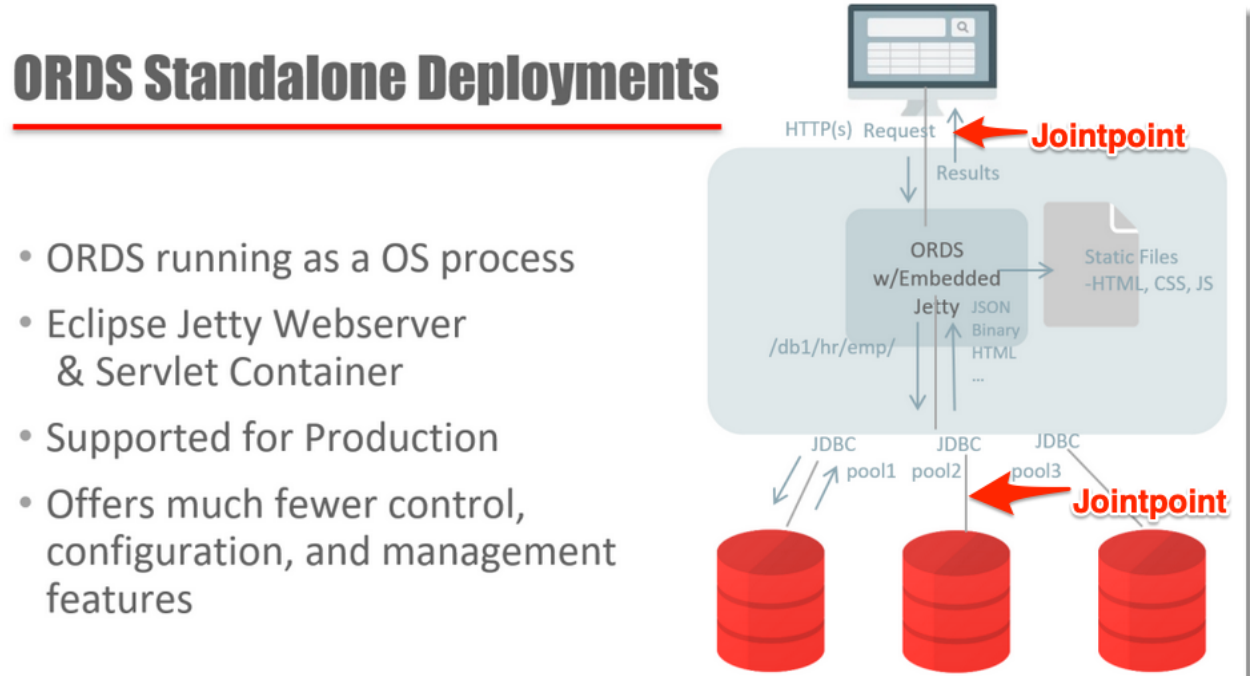
AspectJ and Opentracing for automatic instrumentation and tracing of Java application

AspectJ implements Aspect Oriented Programming (AOP) approach to Java development. AOP supports separation of concerns meaning that a developer can focus on a business logic and service functions like logging, security checks and others can be added separately.

Example of the AspectJ based tracing implementation for the ORDS

Here is below the description of how AspectJ helped to get needed information from Oracle Data Service (ORDS). ORDS is web service allowing to run requests against ORACLE database using REST calls. The ORDS is an ideal show case for the instrumentation because it is compact and includes the main components any Web application would have: application server , database.
The diagram for the standalone RDS deployment taken from https://www.thatjeffsmith.com/archive/2019/02/ords-architecture-a-common-deployment-overview/ shows that request first go to application server (Jetty in this case) and then to the Oracle database.



The points of interest marked in the picture are:

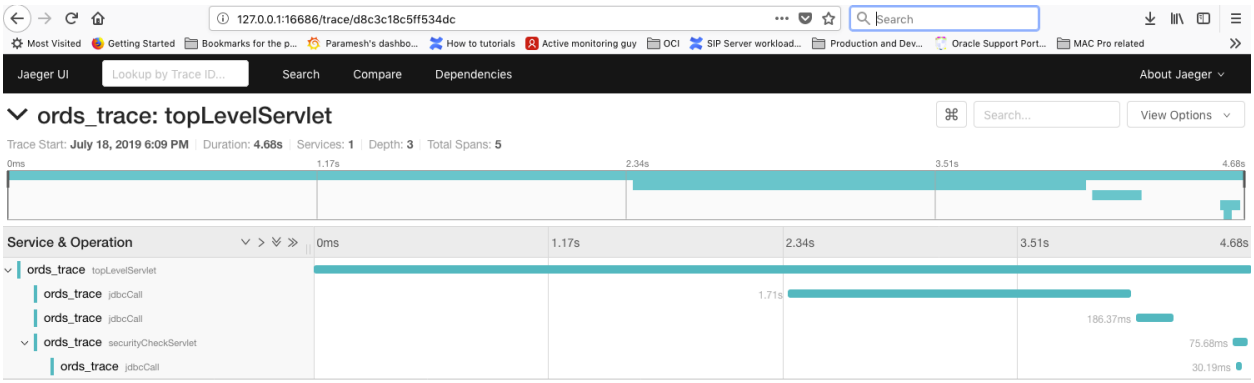
- Total time of the request – time measured on the servlet and then time spent on the JDBC calls to the DB.

These points (joint points in AOP terminology) have been defined in an AspectJ's Aspect and results of the tracing at the joint points are in Jaeger's snapshot.

When these joint pointcuts are reached in the code the respective functions/advices will be executed. The advice could be anything what is needed to get more information. Aspects are the special kind of Java classes which can be added/"weaved" into the original application during compile or load time. So, there are no changes to the application's code.

The advices defined in Aspects for the performance analysis can , for example, print elapsed time of methods to the logs and the same information can be sent to a tracer supporting OpenTracing like Jaeger.

The traces obtained with use of the described approach are in the Jaeger's snapshot below.



Steps to use AspectJ instrumentation and enable the tracing

1. Get the project from the GITHUB.
   Copy pom.xml from https://github.com/yurishkuro/opentracing-tutorial/tree/master/java
   The same pom file is already in this project with name opentracing_pom.xml
2. Create directory opentracing_jars

   SLINETSK-mac:ORDS_INST_TEST slinetsk$ mvn dependency:copy-dependencies -f opentracing_pom.xml -DoutputDirectory=./opentracing_jars

3. Install Oracle DB. For example, the following VM image https://www.oracle.com/technetwork/database/enterprise-edition/databaseappdev-vm-161299.html can be installed in the VirtualBox VM.
4. Configure and test ORDS. Good description is here https://technology.amis.nl/2018/01/22/oracle-rest-data-services-ords/
5. Unzip working ords.war into some directory and copy jar files from opentracing_jars into WEB-INF/lib to have them on a classpath for the ORDS application.
6. Zip all files / directories back into ORDS.war

7. Download the AspectJ installation from
   http://www.eclipse.org/downloads/download.php?file=/tools/aspectj/aspectj-1.9.4.jar .
8. Install AspectJ by executing
   $JAVA_HOME/bin/java -jar aspectj-1.9.4.jar
   and following the instructions in installation program's windows.
   The aspectj will be installed in /var/root/aspectj1.9/
9. Compile aspect including joint points of interest (to get times of JDBC calls, JDBC statement creation, servlet's execution) and create a jar file using the following command

   ajc -source 1.8 -target 1.8 -cp .:$CLASSPATH src/ORDSJaegerAspect.java -outxml -outjar lib/ORDSJaegerAspect1.jar

10. Start Jaeger

    docker run --rm -p 6831:6831/udp -p 6832:6832/udp -p 16686:16686 jaegertracing/all-in-one:1.7 --log-level=debug

11. Execute run.sh to start instrumented ORDS
12. Run curl commands like

    curl --request get http://127.0.0.1:8080/ords/hr/hr/employees/7369

13. Go to http://127.0.0.1:16686 to see the collected traces