

¿Qué ofrece Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
 Ese apoyo que siempre quiso tener...

1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces,
 HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
 Gestor de contenidos (Alfresco)
 Aplicaciones híbridas

Tareas programadas (Quartz)
 Gestor documental (Alfresco)
 Inversión de control (Spring)

Control de autenticación y
 acceso (Spring Security)
 UDDI
 Web Services
 Rest Services
 Social SSO
 SSO (Cas)

JPA-Hibernate, MyBatis
 Motor de búsqueda empresarial (Solr)
 ETL (Talend)

Dirección de Proyectos Informáticos.
 Metodologías ágiles
 Patrones de diseño
 TDD

BPM (jBPM o Bonita)
 Generación de informes (JasperReport)
 ESB (Open ESB)



E-mail:

Contraseña:

Deseo registrar mis datos de acceso

[Inicio](#) [Quiénes somos](#) [Tutoriales](#) [Formación](#) [Comparador de salarios](#) [Nuestro libro](#)
[Charlas](#) [Más](#)

Estás en: [Inicio](#) [Tutoriales](#) AppDynamics Lite, encontrar problemas de rendimiento en aplicaciones Java e...



DESARROLLADO POR:
 [Jose Manuel Sánchez Suárez](#)

Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en [Autentia](#): Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

Catálogo de servicios Autentia



[Anuncios Google](#) [Java](#) [JSP Tutoriales](#) [Tutoriales Diseño](#) [Últimas Noticias](#) [Java](#)

Fecha de publicación del tutorial: 2010-09-10   65

Share | [Regístrate para votar](#)

AppDynamics Lite, encontrar problemas de rendimiento en aplicaciones Java en un entorno de producción.

0. Índice de contenidos.

- 1. Introducción.
- 2. Entorno.
- 3. Instalación.
- 4. Primeros pasos.
- 5. Referencias.
- 6. Conclusiones.

 ¿Quieres trabajar en Autentia o que te ayudemos a encontrar un nuevo trabajo?

 Autentia patrocina un nuevo Coderetreat en Madrid junto con agilismo.es y Eden

 Autentia patrocina el Agile Open Spain 2010

 Disponibles gratis, para el iPhone, las cartas de Planning Poker tipo cómic de Autentia

1. Introducción

En este tutorial vamos a ver cómo llevar a cabo la instalación y unos primeros pasos con una herramienta gratuita para monitorizar y detectar problemas de rendimiento en aplicaciones Java en producción **AppDynamics Lite**. No se trata de una herramienta de profiling, está pensada para detectar respuestas lentas a peticiones web, SQLs con bajo rendimiento y errores en las peticiones; y está orientada a un entorno de producción, donde la monitorización no produzca un impacto en el rendimiento.

En palabras de los creadores: es una herramienta pensada para usar en la primera línea de fuego, se instala en 2 minutos y puedes llegar a detectar errores en muy poco tiempo, 15 minutos o menos.

Como comentaba, es gratuita, en su versión Lite, y existe una versión de pago. La versión Lite tiene una serie de limitaciones:

- solo monitoriza lo sucedido en las últimas dos horas,
- no monitoriza las llamadas a servicios web,
- no detecta Memory Leaks, Deadlocks y Payload/Input Data related errors,
- no dispone de alertas y notificaciones ni una cola de incidencias.

Con la versión Lite, en un principio, nos basta, no somos un departamento de QA.

Solo os digo que si tenéis la ocasión de probarla os va a sorprender su sencillez y el nivel de detalle que es capaz de proporcionar sobre nuestro sistema y un problema de rendimiento concreto.

2. Entorno.

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil MacBook Pro 17' (2.93 GHz Intel Core 2 Duo, 4GB DDR3 SDRAM).
- Sistema Operativo: Mac OS X Snow Leopard 10.6.1
- AppAgentLiteBeta 0.91
- Una aplicación web empresarial con la siguiente arquitectura que corre bajo un Apache Tomcat 6.0.20 y tiene como base de datos un MySQL 5.0.88:
 - Spring 2.5.6
 - Hibernate 3
 - JSF 1.2
 - Apache Tomahawk 1.9

3. Instalación.

La instalación es realmente sencilla, es como **hudson** o **sonar**, que vienen listos para usar.

Lo primero es acceder a la url de descarga: <http://www.appdynamics.com/lite.php>



XI Charla
Autentia - Mule



Histórico de
NOTICIAS

Últimos Tutoriales



VMWare
Workstation,
instalación en un
host Microsoft
Windows



VMWare
Workstation,
ideal para SOHO



Cacoo,
herramienta
colaborativa para
hacer diagramas,
incluso prototipado
de pantallas o UML



Introducción a
CAS



Gestión de
eventos en el
cliente con el
soporte de Ajax de
RichFaces

Últimos Tutoriales
del Autor



Gestión de
eventos en el
cliente con el
soporte de Ajax de
RichFaces



Envío de correo
electrónico con
el soporte de Jboss
Seam.



Creación de
servicios web
RESTful con el
soporte de
RESTeasy de Jboss
Seam.





Got **fires** in production? Want to put them out **FAST?**

the initial Lis
["");
System.out.println("]
0 of the contents of the List are
object references, which
"");
System.out.println("]
contains the same
exact object reference s
System.out.println("]
System.out.println("]
System.out.println("List A
em... on... List
B... List

Download Now >>

By clicking on the "Download Now" button, you agree to accept our [Terms of Service](#).

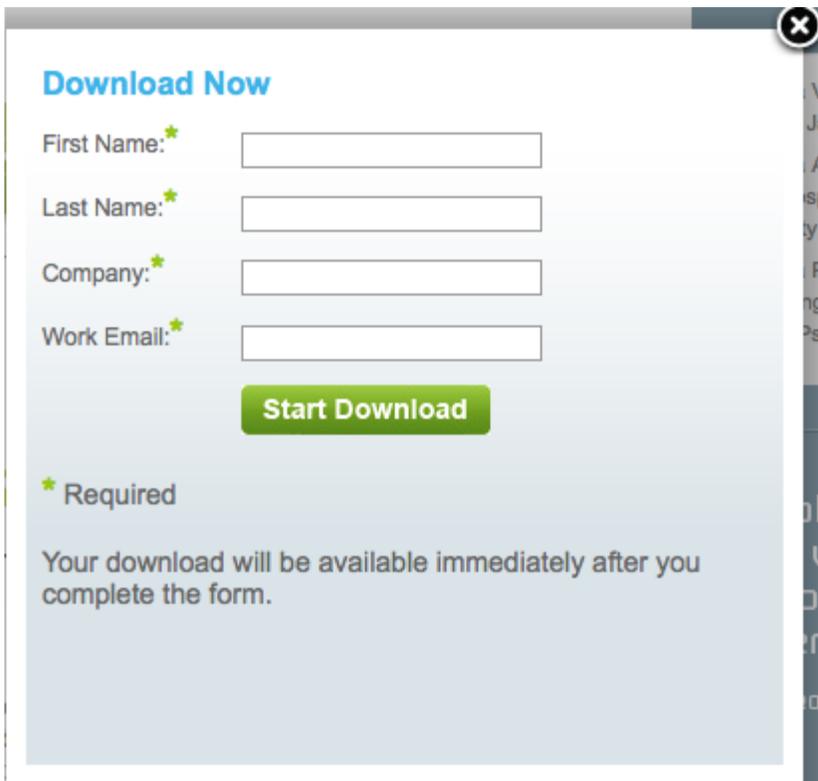


WATCH THE VIDEO: Find out how AppDynamics Lite can help you put out fires—for free.

AppDynamics Lite (beta)

AppDynamics is the very first free product designed for troubleshooting java performance in production environments.

Pulsamos sobre el botón "Download" y se mostrará la siguiente ventana modal:



Download Now

First Name: *

Last Name: *

Company: *

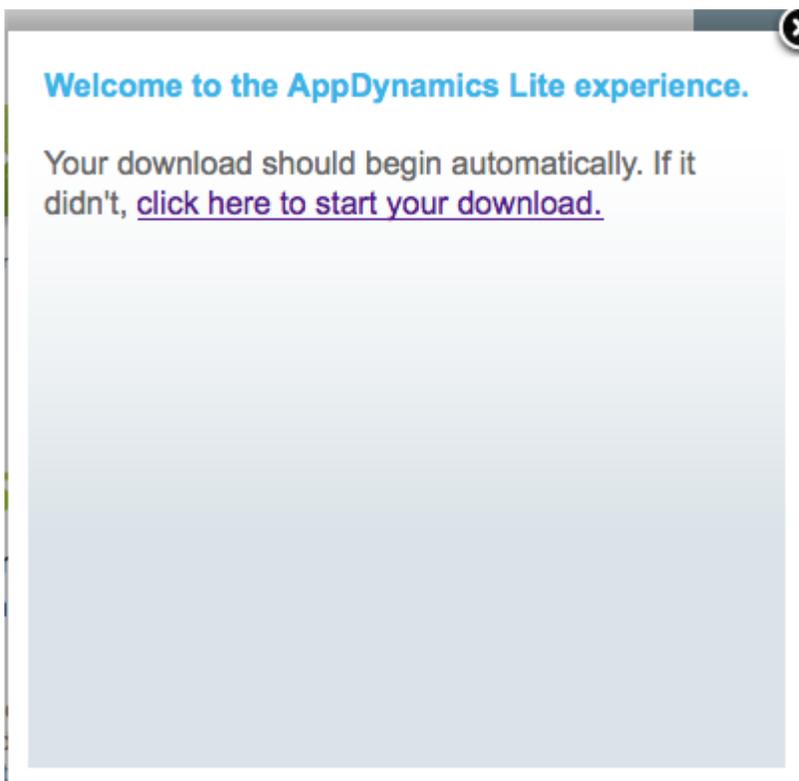
Work Email: *

Start Download

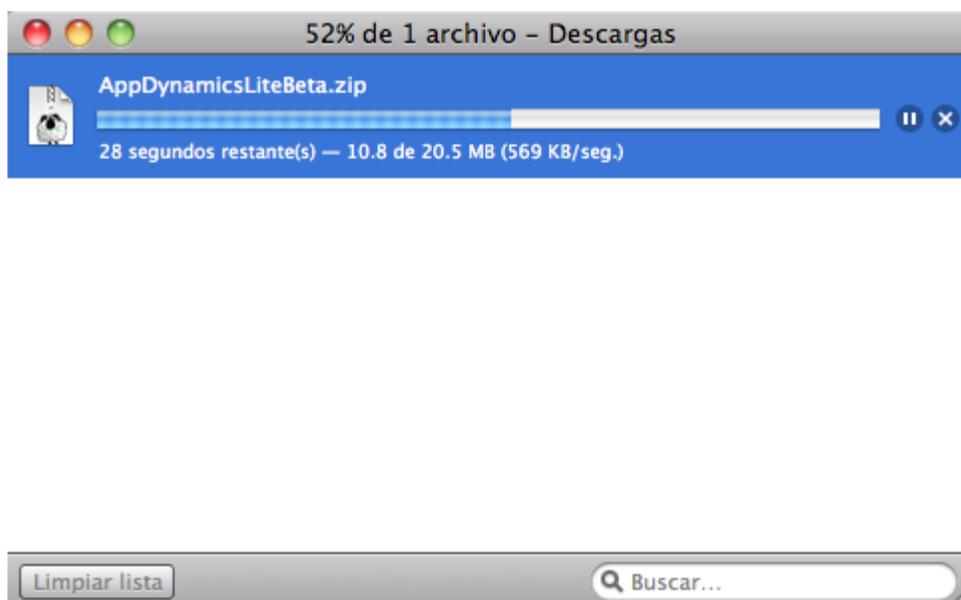
* Required

Your download will be available immediately after you complete the form.

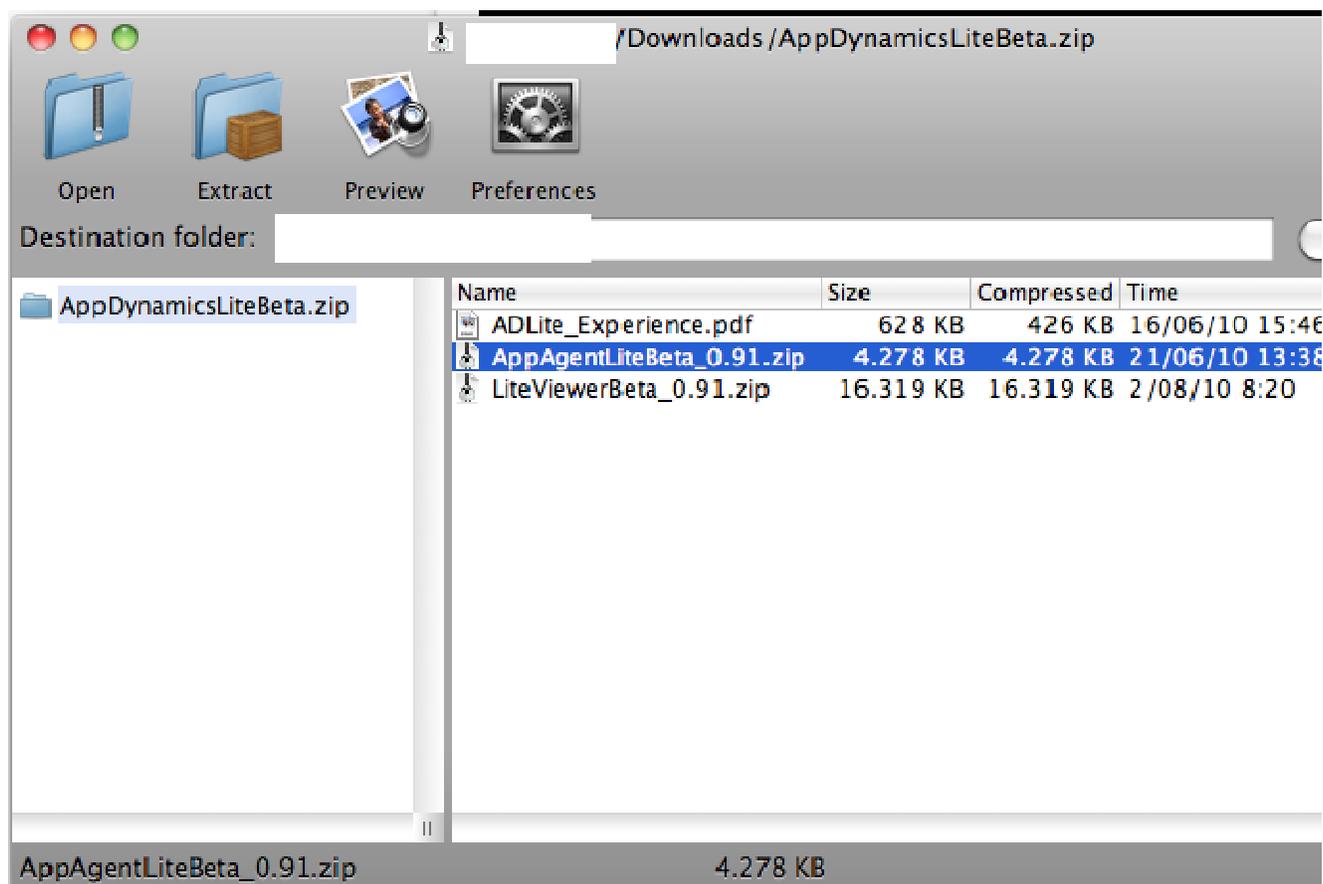
Tras introducir los datos (el email debe ser de empresa), se mostrará una ventana como la que sigue:



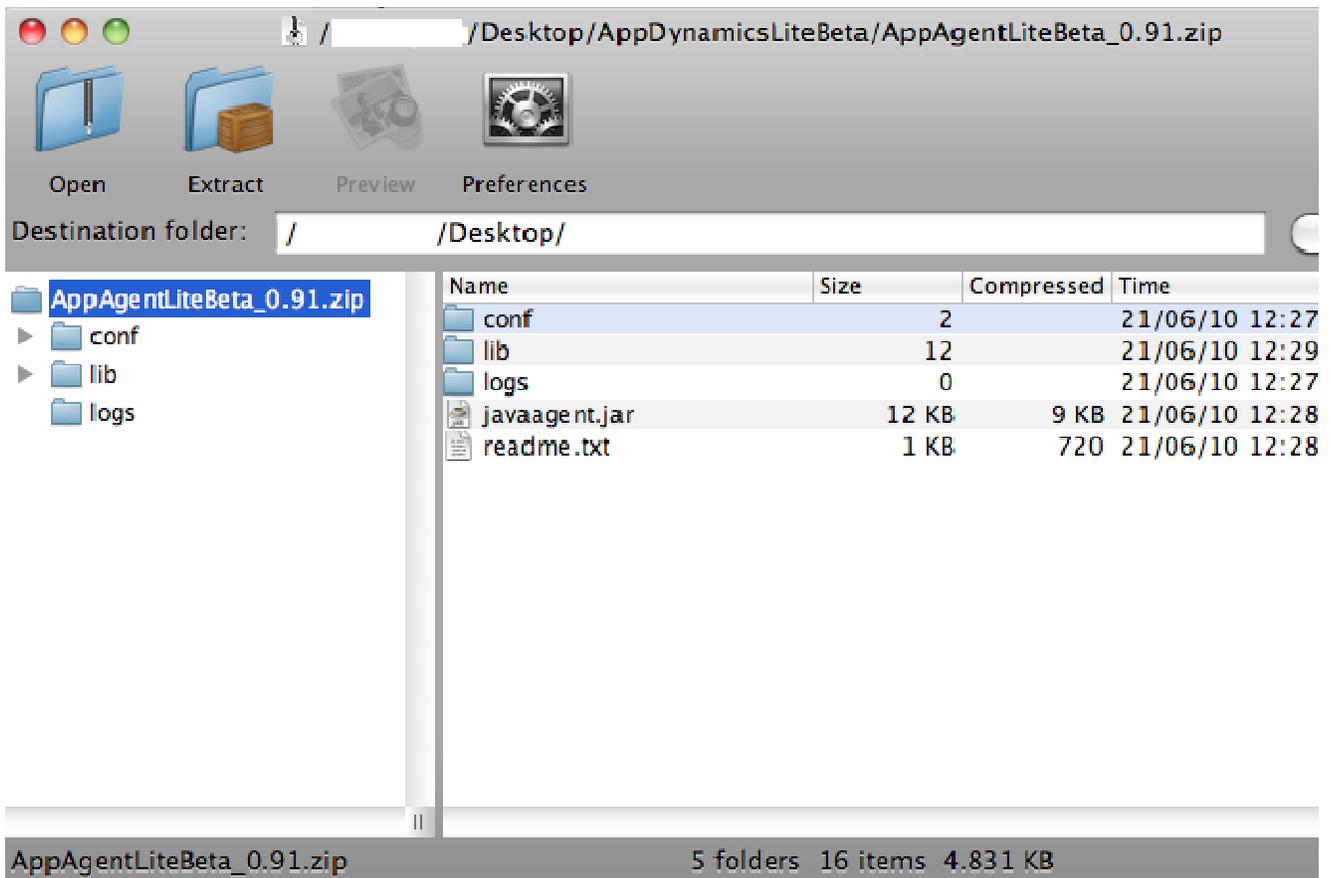
La descarga viene dada o la podemos forzar pulsando en el link, ahora solo depende del tipo de conexión del que dispongamos:



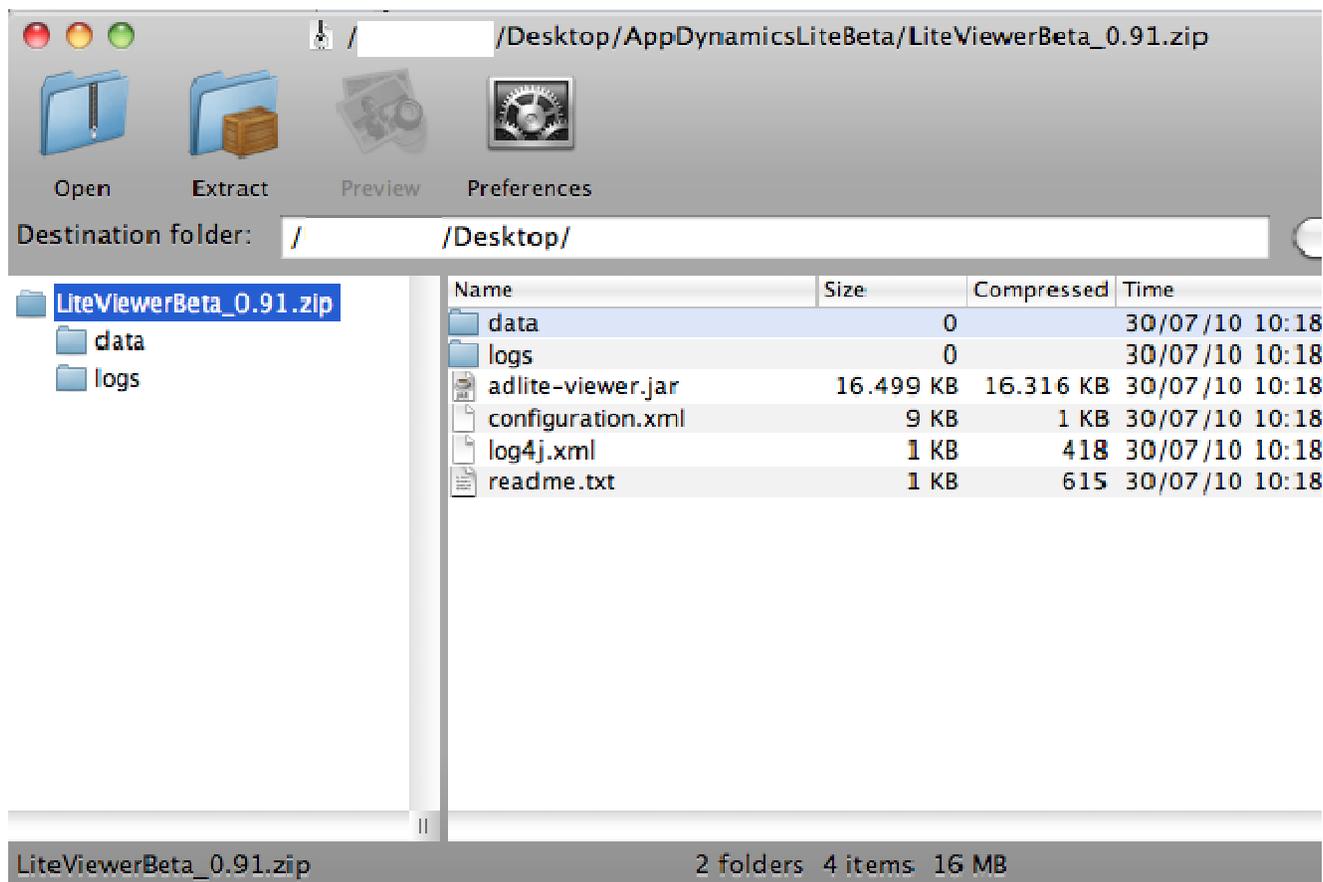
Una vez descargado solo hay que descomprimirlo con tu programa favorito: contiene un pdf con una fact sheet sobre el producto y dos ficheros comprimidos:



El fichero AppAgentLiteBeta_xx.x.zip tiene el siguiente contenido, el objetivo es descomprimirlo en una ruta local.



El fichero LiteViewerBeta_xx.x.zip tiene el siguiente contenido, el objetivo es descomprimirlo en una ruta local.



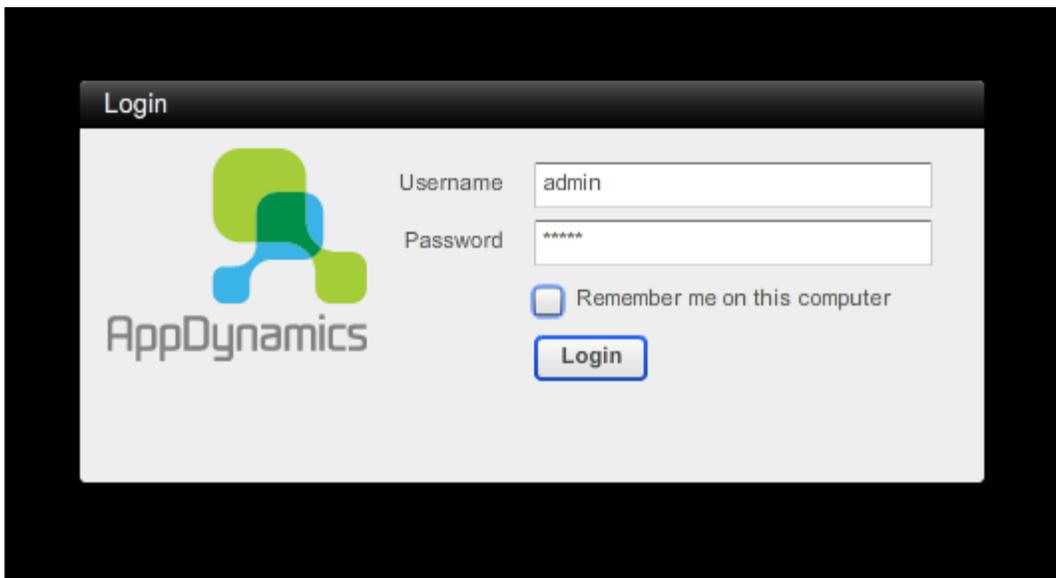
Una vez descomprimido solo falta arrancar ambas piezas:

- el viewer es un visor (una aplicación web con un micronavegador embebido) que recibe y almacena los eventos de monitorización del agente; y
- el javaagent es la pieza que se engancha con la JVM del servidor de aplicaciones e instrumentaliza el código para lanzar los eventos correspondientes: de invocación, tiempo consumido, errores,... Un agente es un interceptor que se pone delante de la invocación a los métodos de las clases que monitoriza y es un estándar a partir de la JDK 5.

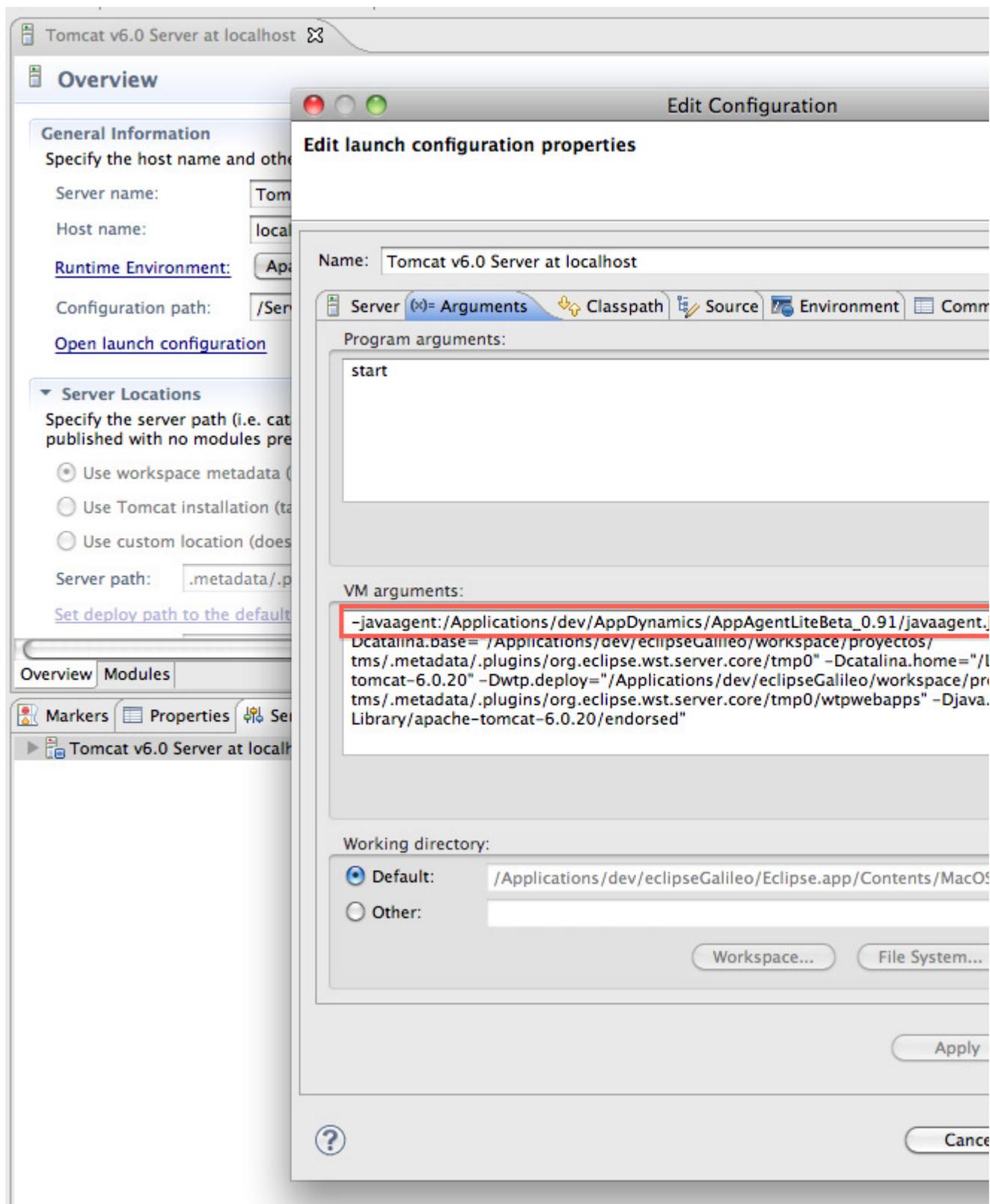
Para arrancar el visor basta con ejecutar el jar `adlite-viewer.jar`, como se muestra en la siguiente imagen, marcándolo como ejecutable en otros SO o por línea de comandos: `java -jar adlite-viewer.jar`.



Por defecto se accede a través del puerto 8990, pero se puede parametrizar y para acceder basta con un navegador y la siguiente url: `http://localhost:8990/`.



El usuario y contraseña es admin:admin y si accedemos sin tener arrancado el agente te informa sobre ello. De momento lo dejamos en segundo plano hasta que hagamos lo siguiente con el agente.



Para registrar el agente basta con introducir el parámetro `-javaagent` con la ruta al fichero `javaagent.jar` en los parámetros de arranque de la máquina virtual. En la imagen anterior se muestra como añadirlo a los parámetros de la VM en la configuración del servidor desde Eclipse; pero se puede añadir al script de arranque del servidor sin problema alguno.

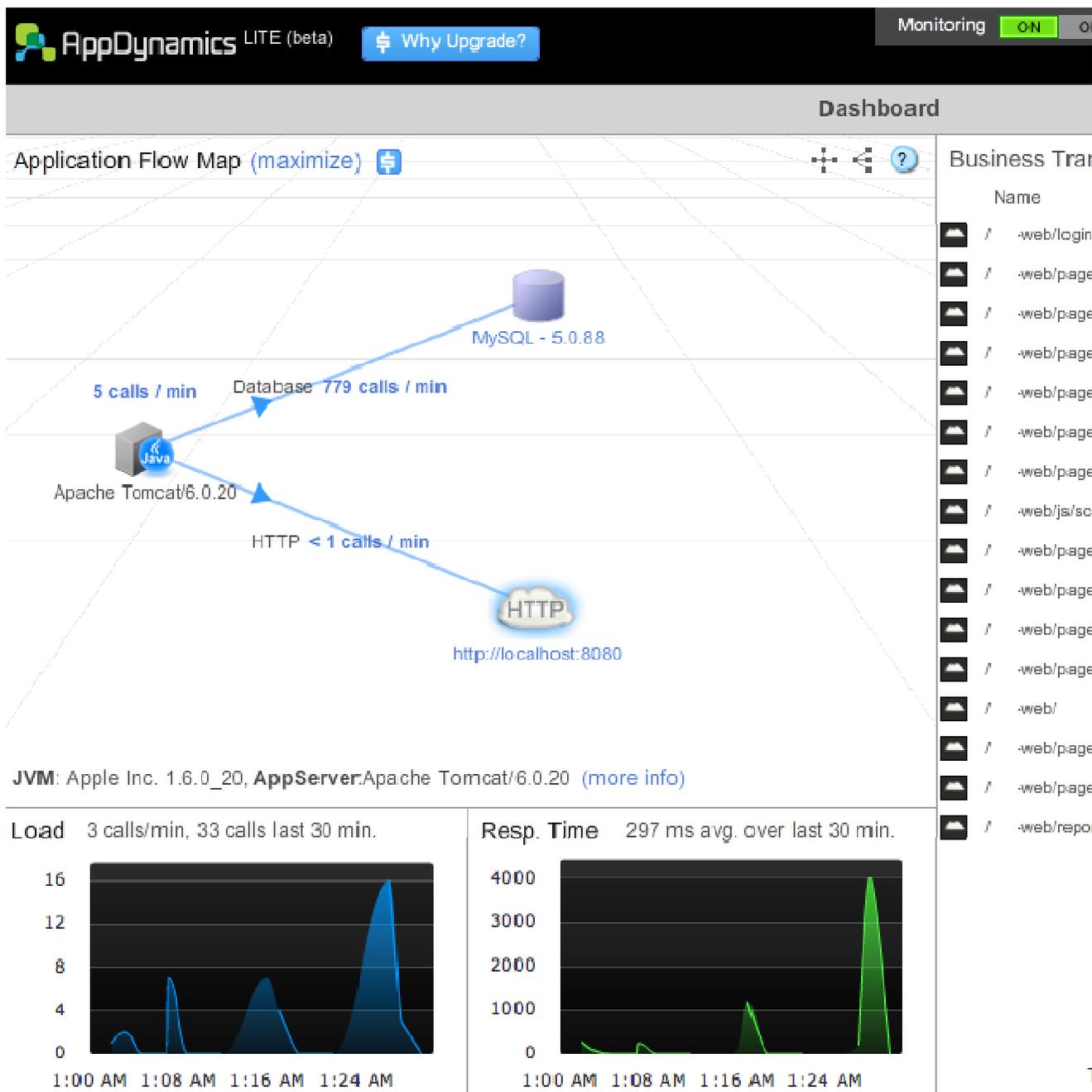
A continuación se muestra la salida por consola del agente nada más arrancar el servidor (como veis en la captura también lo he probado con una aplicación que corre bajo un Jboss Server 4.2.3).

```
JBoss v4.2 at localhost [Generic Server] /System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home/bin/java (0
Install Directory resolved to[/Applications/dev/AppDynamics/AppAgentLiteBeta_0.91]
Using Agent Version [Server Agent Lite v2.1.2.0 GA Build Date 2010-06-21 12:27]
Agent Directory [/Applications/dev/AppDynamics/AppAgentLiteBeta_0.91]
Agent Directory [/Applications/dev/AppDynamics/AppAgentLiteBeta_0.91]
Agent Logging Directory [/Applications/dev/AppDynamics/AppAgentLiteBeta_0.91/logs]
Running obfuscated agent
Registered app server agent with Node ID[1] Component ID[1] Application ID [1]
Started AppDynamics Java Agent Successfully.
```

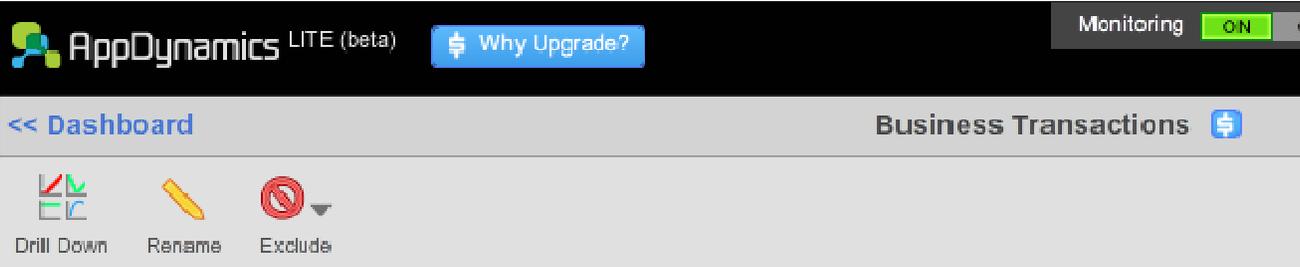
4. Primeros pasos.

Una vez instalado solo queda acceder a la aplicación a monitorizar y navegar un poco por ella. Si sospechamos que alguna parte de la misma adolece de problemas de rendimiento esta es la oportunidad para que salgan a la luz.

Después de generar algo de tráfico, accedemos a la url antes comentada con el usuario y contraseña ya dichos y se mostrará una pantalla similar a la siguiente, ¿no es genial? (en las capturas siguientes he borrado URLs e información sensible de la aplicación sobre la que he realizado las pruebas).



Tenemos un dashboard o cuadro de mando con un esquema de las piezas implicadas en la parte central (servidor, base de datos, invocaciones a servicios externos o internos, a colas de mensajes JMS a Web Services), en la parte derecha, una pila de las llamadas agrupadas por peticiones y ordenadas por rendimiento, y, en la parte inferior, una gráfica de tiempo de carga y respuesta en relación a un periodo de tiempo determinado. Pulsando sobre (maximize) en la parte derecha accedemos a más información sobre la pila de llamadas, lo que denominan transacciones de negocio. Se mostrará una pantalla como la que sigue:



AppDynamics LITE (beta) [Why Upgrade?](#) Monitoring **ON**

<< Dashboard Business Transactions [Why Upgrade?](#)

Drill Down Rename Exclude

Type	Name	Calls	Calls / min	Time (ms)	Errors
Servlet	/web/login.jsf	0	0	0	0
Servlet	/web/p	0	0	0	0
Servlet	/web/p	0	0	0	0
Servlet	/web/p	0	0	0	0
Servlet	/web/p	0	0	0	0
Servlet	/web/p	4	< 1	1342	1
Servlet	/web/p	2	1	199	1
Servlet	/web/j	18	3	2	0
Servlet	/web/p	3	1	240	0
Servlet	/web/p	2	2	602	0
Servlet	/web/p	2	< 1	456	0
Servlet	/web/p	4	2	391	0
Servlet	/web/	1	< 1	0	0
Servlet	/web/p	2	< 1	146	0
Servlet	/web/p	1	< 1	48	0
Servlet	/web/r	1	< 1	12019	0

Haciendo un doble click sobre una fila o pulsando sobre Drill Down, se accede al detalle de la petición, mostrándose una pantalla como la que sigue:

The screenshot displays the AppDynamics Lite interface. At the top, the logo 'AppDynamics LITE (beta)' is visible, along with a 'Why Upgrade?' button and a 'Monitoring ON' indicator. The navigation bar includes links for '<< Dashboard', '<< Business Transactions', and 'Business Transaction Drill Down'. The main content area shows a 'Servlet' endpoint with a cloud icon. Key performance indicators are listed: 'average response time' of 1342 ms, 'calls' of 4, 'calls/min' of 0, and 'stalls' of 0. A 'User Experience' bar is shown with a red segment, indicating '2 normal, 1 slow'. Below this, a 'Captured Bad Requests' section features a 'View Request' button and filter checkboxes for 'Slow', 'Very Slow', 'Error', and 'Stall'. A table lists a single bad request:

User Exp.	Exe. Time	Timestamp	Summary
Slow	2712 ms	09/08/10 01:16:49 AM	Request took [2712], higher than the static threshold of [2000]

En el ejemplo anterior tenemos una petición que ha tardado más de 2 minutos en responder, que la marca con naranja; las peticiones que tardan más de 5 minutos las marca en rojo, así como los errores. Los tiempos son configurables.

Haciendo doble click sobre la fila se accede a una ventana modal con más información sobre la petición, en la que podemos ver los puntos calientes (realizando un filtrado por tiempo consumido).

Slow Request

REQUEST EXECUTION TIME: 2712 ms REQUEST TIMESTAMP: 09/08/10 01:16:49 AM SUMMARY: Request took [2712], higher than the static threshold

Showing 15 of 67 calls. Those taking less than 15 ms have been filtered out.

	Name	Method	Time (ms)	Percentage	External Calls
	com.	.service	1744 ms (self)	64.2 %	JDBC
	com.	.service	232 ms (self)	8.5 %	JDBC
	JSFServlet	.service	105 ms (self)	3.9 %	JDBC
	com.	.service	98 ms (self)	3.6 %	JDBC
	com.	.service	88 ms (self)	3.2 %	JDBC
	com.	.util.lider	70 ms (self)	2.6 %	
	com.	.util.lider	54 ms (self)	2 %	
	com.	.entity.b	44 ms (self)	1.6 %	
	com.	.util.lider	41 ms (self)	1.5 %	
	com.	.util.lider	27 ms (self)	1 %	
	com.	.util.lider	22 ms (self)	0.8 %	
	java.util.concurrent.locks.ReentrantLock		22 ms (self)	0.8 %	

Invocation Trace - some packages have been excluded

(close)

El nivel de detalle es el de la invocación a un método de una clase, y si el mismo ha implicado la ejecución de una sentencia SQL se puede acceder al contenido de la misma desde un enlace que muestra una ventana emergente.

Como se muestra en la siguiente captura también se puede acceder a la pila de sentencias SQL ejecutadas durante la petición, pudiendo visualizar el contenido de las mismas.

Slow Request

<div style="border-bottom: 1px solid #ccc; padding-bottom: 5px;"> REQUEST EXECUTION TIME 2712 ms </div>	<div style="border-bottom: 1px solid #ccc; padding-bottom: 5px;"> REQUEST TIMESTAMP 09/08/10 01:16:49 AM </div>	<div style="border-bottom: 1px solid #ccc; padding-bottom: 5px;"> SUMMARY Request took [2712], higher than the static thre </div>
--	--	--

<div style="border-bottom: 1px solid #ccc; padding-bottom: 5px;"> SUMMARY </div> <div style="padding: 5px 0 5px 10px;"> HOT SPOTS </div> <div style="background-color: #4a7ebb; color: white; padding: 5px 0 5px 10px; margin-bottom: 5px;"> SQL CALLS </div> <div style="padding: 5px 0 5px 10px;"> DATA COLLECTORS + </div> <div style="padding: 5px 0 5px 10px;"> JVM / HARDWARE + </div>	<div style="border-bottom: 1px solid #ccc; padding-bottom: 5px;"> The following SQL calls occurred: </div> <table style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #f2f2f2;"> <th style="font-size: 0.8em;">Type</th> <th style="font-size: 0.8em;">Execution Time (ms)</th> <th style="font-size: 0.8em;">SQL</th> </tr> </thead> <tbody> <tr> <td style="font-size: 0.8em;">Query</td> <td style="font-size: 0.8em;">554</td> <td style="font-size: 0.8em;">SELECT JOB0 .ID AS ID16 14 , JOB0 .CANCELREA CATI COR AS C JOB REFI JOB TRAI TRAI WOF COM COM BILL BUD CATI COM AS C AS C COM HOU INVC DES LAST AS M COM COM REQ COM REV TOT COM TOT</td> </tr> </tbody> </table>	Type	Execution Time (ms)	SQL	Query	554	SELECT JOB0 .ID AS ID16 14 , JOB0 .CANCELREA CATI COR AS C JOB REFI JOB TRAI TRAI WOF COM COM BILL BUD CATI COM AS C AS C COM HOU INVC DES LAST AS M COM COM REQ COM REV TOT COM TOT
Type	Execution Time (ms)	SQL					
Query	554	SELECT JOB0 .ID AS ID16 14 , JOB0 .CANCELREA CATI COR AS C JOB REFI JOB TRAI TRAI WOF COM COM BILL BUD CATI COM AS C AS C COM HOU INVC DES LAST AS M COM COM REQ COM REV TOT COM TOT					

[\(close\)](#)

Por último, desde la parte inferior derecha de la ventana de detalle de la petición (sampled snapshot) se puede acceder al detalle de una serie de peticiones de muestreo, pudiendo visualizar, como se muestra a continuación, la pila de ejecución de la invocación a un servicio, con un ratio del tiempo consumido en cada uno de los métodos.

Snapshot Execution Time: 2716 ms | Snapshot Timestamp: 01:16:49 AM | Reason: First Snapshot for Occurrence Sampling. Event

CALL GRAPH

Go to root | Go up one level | Callgraph navigation help

Name	Time (ms)
com.autentia.common.util.PagedList: get:146	0 ms (self)
com.autentia.common.util.PagedList: checkLoadPage:126	0 ms (self)
com.autentia.common.util.PagedList: loadPage:138	0 ms (self)
com.service.GenericDao\$2: getP	0 ms (self)
com.service.GenericDao: acc	0 ms (self)
com.service.GenericDao:	1744 ms (self)
com.entity.Commission	0 ms (self)
com.util.Identifier: g	27 ms (self)
com.entity.Commission	0 ms (self)
com.util.Identifier: g	22 ms (self)
com.entity.Commission	0 ms (self)
com.util.Identifier: g	11 ms (self)
com.entity.Commission	0 ms (self)
com.util.Identifier: g	54 ms (self)
java.util.concurrent.locks.ReentrantLock	22 ms (self)
java.util.concurrent.locks.ReentrantLock	22 ms (self)
com.entity.Commission	0 ms (self)
com.util.Identifier: g	5 ms (self)
com.entity.billing.Cos	44 ms (self)
java.ReentrantLock	18 ms (self)

(close) * Some packages have been excluded from this Call Graph

Pulsando sobre una fila se puede acceder a más detalle sobre la invocación a un método.

Snapshot Execution Time: 2716 ms | Snapshot Timestamp: 01:16:49 AM | Reason: First Snapshot for Occurrence Sampling. Ev...

CALL GRAPH navigation: Go to root, Go up one level, Callgraph navigation help

Name	Time (ms)
com.autentia.common.util.PagedList:get:146	0 ms (self)
com.autentia.common.util.PagedList:checkLoadPage:126	0 ms (self)
com.autentia.common.util.PagedList:loadPage:138	0 ms (self)
com.:.GenericDao\$2:getIP	0 ms (self)
com.:.GenericDao:acc	0 ms (self)
com.:.service.GenericDao:findAllByQueryAndCount	1744 ms (self)
com.:.Commission	0 ms (self)
.util.Identifier:c	5 ms (self)
tity.billing.Cos	44 ms (self)
2entrantLock	18 ms (self)

Execution Time: 1744 ms (self) (?), 2024 ms (total) (?). Made 2 JDBC external call(s). (close)

(close) * Some packages have been excluded from this Call Graph (?)

Sorprende que con tan poco configurado se tenga accesible tanta información y funcionando a la primera.

Se puede pinchar la invocación a los siguientes componentes, los beans de Spring no vienen por defecto seleccionados.

AppDynamics LITE (beta) [Why Upgrade?](#)
Monitoring ON

<< [Dashboard](#)
Configuration

Transaction Detection
Request Thresholds
Snapshot Sampling
Error and Stall Detection

▼ Business Transaction Identification Configuration

This page helps you understand and configure how Business Transactions are identified and named. All configurations are automatically discovered, or matched against Custom Match Rules. Custom Match rules take precedence over automatic detection.

Type	Transaction Monitoring	Automatic Transaction Detection
 Servlet	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for all Servlet requests, and name them. Click here to use only part of the URI to name Transactions
 Struts Action	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for all Struts Action invocations, and name them.
 Web Service	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for all Web Service requests, and name them.
 POJO	<input checked="" type="checkbox"/> Enabled	Any java method can be the entry point for a Business Transaction. The configuration parameters like its name, its super class name, the interfaces it implements, etc.
 EJB	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/> Discover Transactions automatically for all EJB invocations, and name them.
 Spring Bean	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for all Spring Bean invocations, and name them.
 JMS	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for all incoming JMS Messages, and name them.

Y se puede configurar el nivel de detalle de las SQL, si necesitamos conocer el valor de los parámetros de las sentencias o queremos que lo filtre con ?.

AppDynamics LITE (beta) [Why Upgrade?](#)
Monitoring ON

<< [Dashboard](#)
Configuration

Transaction Detection
Request Thresholds
Snapshot Sampling
Error and Stall Detection

com.sun.*	Sun Reference Implementations/JDK Core
sun.*	Sun Reference Implementations/JDK Core
org.*	Common Open Source Libraries like
com.bea.*	BEA Core

Add Custom Package Exclude
Remove Selected

Specific subpackages / classes to from the Excluded Packages to be included in Call Graphs

To force inclusion of specific classes or sub packages from the Excluded Packages list above, specify them below can force inclusion of calls to java.util.HashMap by adding it to the list below.

Packages/Classes Always Shown	Description

Add Always Shown Package/Class
Remove Selected

SQL Capture Settings

Capture Raw SQL - This will capture raw SQL statements with parameter values included. For example "select * from

Filter Parameter Values - This will filter out all parameter values from the captured SQL. For example "select * from u

5. Referencias.

- <http://blog.codecentric.de/en/2010/08/troubleshoot-java-in-production-introducing-appdynamics-lite/>
- <http://blog.codecentric.de/en/2010/08/easy-performance-analysis-with-appdynamics-lite/>
- <http://www.appdynamics.com/lite.php>

6. Conclusiones.

Para mi, una herramienta a tener en cuenta en caso de emergencia. Sabemos que en el entorno de desarrollo no se suele disponer del mismo juego de datos que en el

entorno de producción y nos enfrentamos con problemas de rendimiento más veces de las que desearíamos. Poder disponer de una herramienta como esta nos puede sacar de un apuro y nos ayudará a la realización de una primera auditoría de rendimiento.

Como decía, existen otras herramientas para realizar pruebas de carga, no es el objetivo de AppDynamics Lite; pero ante un error puntual de la aplicación o una deficiencia en el rendimiento, podemos pinchar directamente en producción y llevarnos una información muy valiosa sobre el comportamiento real del sistema, con una configuración mínima.

Espero que lo disfrutéis.

Un saludo.

Jose

jmsanchez@autentia.com

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

» **Registrate** y accede a esta y otras ventajas «

COMENTARIOS



franferri

2010-09-10 - 09:36:01

*Se parece mucho al wily introscope para java (t.b. .Net)
Es como una modernización de la parte de medición de rendimiento.
Muy útil.*



Esta obra está licenciada bajo [licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)

