

¿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)

Estás en: Inicio » Tutoriales » OSCache: Sistema de caché para aplicaciones Java

	<p>DESARROLLADO POR:</p> <p>Carlos García Pérez</p>	<p>Técnico especialista en informática de empresa (CEU). Ingeniero Técnico en Informática de Sistemas (UPM) Creador de MobileTest, Haaala!, Girillo, toi18n. Charla sobre desarrollo de aplicaciones en Android. Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación</p>
--	--	--



Fecha de publicación del tutorial: 2009-01-02

5.438

Share |

Regístrate para votar

OSCache: Sistema de caché para aplicaciones Java

Índice de contenido

- Introducción.
- Caso de estudio: Aplicación de escritorio.
- Caso de estudio: Aplicación Web.
- Referencias a otros sistemas de caché Open Source
- Conclusiones

Introducción

Normalmente hay secciones y datos dentro de nuestras aplicaciones que no varían con frecuencia o cuya frecuencia está controlada y es conocida. Además, obtener esta información es costosa en tiempo o recursos informáticos. pues bien, es obvio que si cacheamos esta información el rendimiento de nuestra aplicación incrementará y por lo tanto la satisfacción de los usuarios que la utilizan también mejorará.

En este tutorial vamos a ver uno de los sistemas de caché más extendidos y aceptados, se trata de OSCache

Caso de estudio: Aplicación de escritorio

A continuación, vamos a programar una sencilla aplicación de escritorio en donde aprenderemos a realizar las tareas de cacheo, descacheo y acceso a la información cacheada. Se trata de una sencilla aplicación en donde cada operación se realiza al hacer clic en un botón, y en donde se mostrará el tiempo que transcurre desde que se inicia la tarea hasta que finaliza. Para ello, se simula el acceso a una fuente de datos cuyo costo en tiempo es bastante elevado, por lo que decidimos cachearlo.

La intención de esta sencilla aplicación es que aprenda y observe lo que va sucediendo al acceder a datos cacheados, acceder a datos que no han sido cacheados, etc.



Configuración de OSCache para este ejemplo

OSCache se configura fácilmente a través de un archivo de configuración de nombre `oscache.properties`. Aunque para usarlo no hay por que tocar nada, en este tutorial vamos a cambiar el comportamiento de por defecto (cache en memoria sin limite en el número de elementos a cachear) para habilitar el cacheo en disco, de manera que si la aplicación se cierra no haya que volver a cachear los datos.

Desde mi punto de vista las propiedades que he modificado son las más importantes, así que observe los comentarios para conocer su significado.

Habilitamos el cache en disco

```
cache.persistence.class=com.opensymphony.oscache.plugins.diskpersistence.DiskPersistenceListener
```

Especificamos el directorio donde se cacheará la información

```
cache.path=c:\\oscache_temp
```

Indicamos que queremos un límite también para los elementos cacheados en disco

```
cache.unlimited.disk=false
```

Como mucho serán cacheados 500 objetos simultáneamente

```
cache.capacity=500
```

Cuando se supere el límite de 500 objetos, se aplicará el algoritmo LRU (Least Recently Used) para obtener espacio en disco y cachear los nuevos elementos.

```
cache.algorithm=com.opensymphony.oscache.base.algorithm.LRUCache
```

Para más información dirígame a la documentación de configuración oficial.

Código fuente de la aplicación:

La siguiente clase simula un acceso lento (2 segundos) para obtener una información que luego será cacheada.

```
view plain print ?
01. package com.autentia.tutoriales.oscache;
02.
03. /**
04.  * Gestión de clientes.
05.  * Simula accesos lentos a los datos.
06.  * @author Carlos García. Autentia.
07.  * @see http://www.mobiletest.es
08.  */
09. public class CustomersCtrl {
10.
11.     private CustomersCtrl(){}
12.
13.     /**
14.      * @return Devuelve todos los clientes
15.      */
16.     public static java.util.ArrayList<String> getAll(){
17.         java.util.ArrayList<String> customers = new java.util.ArrayList<String>();
18.
19.         for (int i = 0; i < 5; i++){
20.             customers.add("Cliente " + String.valueOf(i));
```

Catálogo de servicios Autentia



Últimas Noticias

- VII Autentia Cycling Day
- Autentia patrocina la charla sobre Java SE 7 en Madrid
- Alfresco Day 2011
- XVII Charla Autentia - Grails - Vídeos y Material
- iii 15 millones de descargas de tutoriales !!!

Histórico de NOTICIAS

Últimos Tutoriales

- Spring MVC: acceder a las propiedades de un fichero desde una JSP con Expression Language (EL)
- Framework Scala liftweb
- Trabajando con JAXB y Eclipse
- Configurar Spring Security 3.1 para autenticarse contra un Active Directory
- Migración a ICEfaces 2.0

Últimos Tutoriales del Autor

- Construcción de un control personalizado en Android
- Desarrollo de aplicaciones mixtas (web/nativa) en Android
- toi18n, Traduce tus aplicaciones de forma rápida Online
- Haaala! para Android: Facebook y Email con Voz
- Girillo TTS: Una aplicación Android para evitar el uso del móvil durante la conducción

Síguenos a través de:



Últimas ofertas de empleo

- 2011-07-06 Otras Sin catalogar - LUGO.
- 2011-06-20 Comercial - Ventas - SEVILLA.
- Contabilidad - Especialista Contable - BARCELONA.
- 2011-05-14 Comercial - Ventas - TARRAGONA.
- 2011-04-13 Comercial - Ventas - VALENCIA.

```

21.     }
22.     // Retrasamos la ejecución, simulando una operación costosa en tiempo
23.     try {
24.         Thread.sleep(2000);
25.     } catch (java.lang.InterruptedExcepcion ex){
26.         // No hacemos nada.
27.     }
28.
29.     return customers;
30. }
31. }
32. }
33.
34.

```

La siguiente clase representa una ventana con distintos botones cuyas tareas estarán realiccionadas con operaciones con la caché.

```

view plain print ?
01. package com.autentia.tutoriales.oscache;
02.
03. import java.awt.*;
04. import java.awt.event.ActionEvent;
05. import java.util.ArrayList;
06.
07. import javax.swing.*;
08.
09. import com.opensymphony.oscache.general.GeneralCacheAdministrator;
10. import com.opensymphony.oscache.web.filter.ExpiresRefreshPolicy;
11. import com.opensymphony.oscache.base.NeedsRefreshException;
12.
13. /**
14.  * Ventana para probar el API de OSCache
15.  * @author Carlos Garcia. Autentia.
16.  * @see http://www.mobilitytest.es
17.  */
18. public class OSCacheTestFrame extends JFrame implements java.awt.event.ActionListener {
19.
20.     private static final String ALL_CUSTOMER_CACHE_KEY = "CUSTOMERS";
21.
22.     // Atributos funcionales
23.     private GeneralCacheAdministrator cache;
24.     private ArrayList<String> customers;
25.
26.     // Atributos gráficos
27.     private JLabel lblTime;
28.     private JPanel contentPane;
29.     private JPanel buttons;
30.     private JButton btnAppendDataToCache;
31.     private JButton btnReadFromCache;
32.     private JButton btnDeleteCache;
33.
34.     /**
35.      * Constructor
36.      */
37.     public OSCacheTestFrame(){
38.         super("Pruebas con OSCache");
39.
40.         cache = new GeneralCacheAdministrator();
41.
42.         this.createUI();
43.     }
44.
45.     /**
46.      * Crea el interfaz gráfico
47.      */
48.     private void createUI() {
49.         try {
50.             UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
51.         } catch (Exception ex) {
52.             //No se dara
53.         }
54.
55.         lblTime = new JLabel();
56.         contentPane = new JPanel(new BorderLayout());
57.         buttons = new JPanel(new FlowLayout());
58.
59.         btnAppendDataToCache = new JButton("Cachear datos");
60.         btnReadFromCache = new JButton("Leer datos");
61.         btnDeleteCache = new JButton("Borrar cache");
62.
63.         contentPane.add(lblTime, BorderLayout.NORTH);
64.         contentPane.add(buttons, BorderLayout.SOUTH);
65.
66.         buttons.add(btnReadFromCache);
67.         buttons.add(btnAppendDataToCache);
68.         buttons.add(btnDeleteCache);
69.
70.         btnAppendDataToCache.addActionListener(this);
71.         btnReadFromCache.addActionListener(this);
72.         btnDeleteCache.addActionListener(this);
73.
74.         this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
75.         this.setSize(new Dimension(400, 200));
76.         this.setContentPane(contentPane);
77.     }
78.
79.     /* Tareas de Liberación de recursos
80.      * @see java.awt.Window#dispose()
81.      */
82.     public void dispose() {
83.         super.dispose();
84.         cache.destroy();
85.     }
86.
87.     /**
88.      * Intenta Leer Los datos de La caché. En caso de que no exista Los Lee de La "base de d
89.      */
90.     private void readFromCacheTask() {
91.         try {
92.             customers = (ArrayList <String>) cache.getFromCache(ALL_CUSTOMER_CACHE_KEY);
93.         } catch (NeedsRefreshException e) { // Los datos de La cache no existen o han caduca
94.             cache.cancelUpdate(ALL_CUSTOMER_CACHE_KEY);
95.             customers = CustomersCtrl.getAll();
96.         }
97.     }
98.
99.     /**
100.      * Añade datos a La cache con un período de validez de 120 segundos
101.      */
102.     private void putDataOnCacheTask(){
103.         customers = CustomersCtrl.getAll();
104.         cache.putInCache(ALL_CUSTOMER_cache_KEY, customers, new ExpiresRefreshPolicy(120));
105.     }
106.
107.     /**
108.      * Vacía La cache
109.      */
110.     private void deleteCacheTask() {
111.         cache.removeEntry(ALL_CUSTOMER_CACHE_KEY);
112.     }
113.
114.     /*
115.      * @see java.awt.event.ActionListener#actionPerformed(java.awt.event.ActionEvent)
116.      */
117.     public void actionPerformed(ActionEvent evt) {
118.         Object source = evt.getSource();

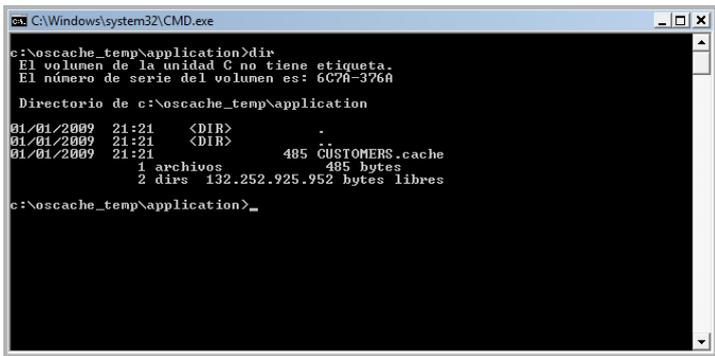
```

```

119.     boolean isError = false;
120.     long   startTime = System.currentTimeMillis();
121.
122.     try {
123.         // Todas las tareas deberían realizarse en un hilo independiente para no
124.         // bloquear el hilo principal de repaintado
125.         if (source == btnReadFromCache){
126.             readFromCacheTask();
127.         } else if (source == btnDeleteCache){
128.             deleteCacheTask();
129.         } else if (source == btnAppendDataToCache){
130.             putDataOnCacheTask();
131.         }
132.     } catch (Exception ex){
133.         isError = true;
134.         JOptionPane.showMessageDialog(this, ex.getMessage());
135.     } finally {
136.         if (!isError){
137.             lblTime.setText("Tiempo en realizar la tarea " + (System.currentTimeMillis()
138.                 - startTime) +
139.                 " milisegundos");
140.         }
141.     }
142.
143.     /**
144.      * Inicia el programa de testeo del API OSCache
145.      */
146.     public static void main(String[] args) {
147.         JFrame appWin = new OSCacheTestFrame();
148.         appWin.setVisible(true);
149.     }
150.
151.

```

Si observa la configuración de OSCache para esta aplicación, la caché en disco está habilitada, de manera que en la siguiente imagen se puede observar como OSCache crea y destruye los archivos en donde guarda la información cacheada:



Nota: Al estar habilitada la cache en disco, si cierra la aplicación y la vuelve abrir verá que los datos siguen estando cacheados, es decir, los tiempos de lectura tienden a cero. (En este caso la cache tiene un tiempo de validez de 120 segundos desde que fué cacheada).

Caso de estudio: Aplicación Web.

OSCache, además de ser un sistema de cache de propósito general sobre el cuál podemos cachear objetos. Incluye una librería de Tags y un filtro que proporcionan de forma sencilla la posibilidad de cachear secciones o páginas completas de JSP de nuestras aplicaciones Web.

Si desea más información al respecto, le recomiendo que lea el siguiente tutorial. [Ver el tutorial.](#)

Referencias a otros sistemas de caché Open Source

El siguiente enlace se enumeran otros sistemas de caché Open Source:

[Open Source cache Solutions in Java](#)

Conclusiones

Bueno como veis es bastante sencillo de integrar en nuestras aplicaciones y las mejoras de rendimiento pueden ser notables. Eso sí, para la gente nueva en este tema, conviene profundizar más al respecto y elegir bien los puntos estratégicos a cachear.

Carlos García Pérez. Creador de MobileTest, un complemento educativo para los profesores y sus alumnos. cgpcosmad@gmail.com

Animáte 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.

(Sólo para usuarios registrados)

» [Regístrate](#) y accede a esta y otras ventajas «

COMENTARIOS



Daniel Alonso

2009-01-14 - 01:02:08 PM

Artista! Que soy Dani, el de IECI :D. Muy bueno el tutorial, a ver si le echo un vistazo ^_^ A cuidarse ;D

 Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5