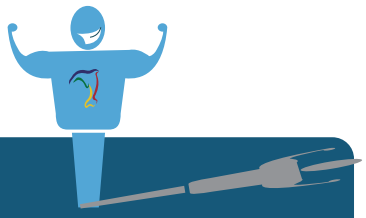
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

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

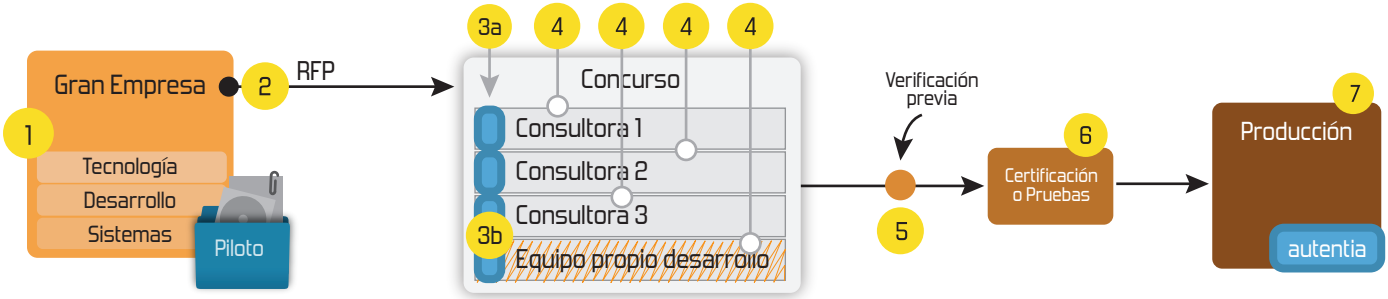
1. Desarrollo de componentes y proyectos a medida



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

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

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

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

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Tutorial desarrollado por

Carlos García Pérez

Creador de MobileTest, un complemento educativo para los profesores y sus alumnos.

Consultor tecnológico en el desarrollo de proyectos informáticos.

Ingeniero Técnico en Informática *

Puedes encontrarme en Autentia

Somos expertos en Java/J2EE

Descargar este documento en formato PDF: jtapi.pdf

Fecha de creación del tutorial: 2009-04-13

JTAPI. Telefonía sobre Java

En mi vida profesional estuve varios años trabajando en el departamento de investigación y desarrollo de una empresa de telefonía. En este tutorial, voy a brindar al lector con conocimientos básicos de JTAPI una serie de ejemplos que le orienten y sirvan de base para realizar sus pruebas en su plataforma telefónica.

Este tutorial no es teórico, en caso de querer aprender acerca de este tema, recomiendo que se lea la especificación pues es bastante clara y fácil de comprender.

Introducción

En el mundillo Java, JTAPI es a la telefonía lo que JDBC es a las bases de datos.

Al igual que JDBC, JTAPI **es una especificación (no una implementación)** definida por Sun para el desarrollo de aplicaciones CTI en el lenguaje de programación Java.

¿Para que sirve JTAPI?

Su principal uso se da en aquellos entornos donde se requiere una programación de sistemas telefónicos. Por ejemplo, en los **centros de llamadas** a través de los cuales las empresas realizan tareas de atención al cliente, telemarketing, etc.

¿Qué puedo hacer con JTAPI?

Entre otras muchas cosas podrá hacer las siguientes tareas:

1. Realizar llamadas.
2. Contestar llamadas.
3. Capturar llamadas que están sonando en un determinado terminal.
4. Realizar conferencias.

5. Realizar transferencias tanto directas como indirectas.
6. Realizar llamadas predictivas.
7. Retener llamadas.
8. Redireccionar llamadas.
9. Recuperar llamadas retenidas.
10. Obtener información de las llamadas en curso.
11. Asociar datos a llamadas.
12. Gestionar grupos de teleoperadores (agentes).
13. Logar/deslogar a agentes en grupos.
14. Gestionar las rutas por las que pasará la llamada.
15. Gestionar la pantalla y el teclado de los dispositivos telefónicos.

Estructuración de JTAPI

JTAPI está estructurado en los siguientes paquetes:

1. Core
2. CallControl
3. CallCenter
4. Phone

Paquete Core

Bajo este paquete se incluyen las clases que aportan la funcionalidad básica que **debe implementar** un sistema telefónico para decir que implementa JTAPI.

1. Realizar, finalizar y contestar llamadas.
2. Obtener el estado en el que se encuentran el proveedor, las llamadas, los componentes de la llamada.
3. Obtener los terminales y direcciones que forman parte del dominio del entorno telefónico.
4. Poner oyentes a las llamadas, terminales, proveedor, etc. para que se nos avise mediante la invocación de un método que ha sucedido algo. Por ejemplo la llamada a finalizado, o el proveedor ha dejado de estar en servicio.

Ejemplo: Realización de una llamada telefónica.

2009-03-27
Primeros pasos con Moodle

2009-03-26
Introducción a JSF Java

2009-03-25
A1 Website Analyzer

2009-03-24
Cómo ver el correo de Gmail sin conexión a Internet

2009-03-20
JasperReports Maven Plugin

2009-03-16
Creación de contenidos SCORM: eXe

2009-03-15
Spring WS: Creación de Servicios Web con Spring

2009-03-13
Instalación Alfresco (Labs)

2009-02-26
Maven JXR Plugin: publica el código fuente en el site

2009-03-15
Generación de XML Schema (XSD) y DTD a partir de documentos XML

2009-03-04
Persistencia con Spring

2009-02-26
Vistas materializadas

2009-02-03
Instalación de MySQL 5.1 en Windows

2009-03-03
Instalación de Java Virtual Machine

2009-03-03
Primeros Pasos con Liferay 5.2.1

2009-02-27
Edicion de video MPEG2

```
view plain  print  ?
01. package com.autentia.tutoriales.jtapi;
02.
03. import javax.telephony.*;
04.
05. /**
06.  * Realiza una llamada telefónica desde un origen a un destino.
07.  * @author Carlos García. Autentia
08.  */
09. public class MakeCallApp {
10.
11.     /**
12.       * @param args Origen y destino de la llamada
13.       */
14.     public static void main(String[] args){
15.         Provider prov = null;
16.
17.         // Verificamos los argumentos. Deben ser dos números de teléfono.
18.         if ( args.length != 2 ){
19.             System.out.println("Formato:\"MakeCallApp <origen> <destino>\"");
20.             return;
21.         }
22.
23.         try {
24.             // Obtenemos una instancia que implemente javax.telephony.JtapiPee
25.             JtapiPeer peer = JtapiPeerFactory.getJtapiPeer(null);
26.
27.             // Obtenemos un proveedor que de el servicio
28.             prov = peer.getProvider(null);
29.
30.             //  Instanciamos y configuramos los objetos necesarios para la lla
31.             Terminal term = prov.getTerminal(args[0]);
32.
33.             // Creamos un objeto llamada
34.             Call call = prov.createCall();
35.
36.             // Realizamos una llamada
37.             call.connect(term, term.getAddresses()[0], args[1]);
38.
39.         } catch ( Exception e ){
40.             System.out.println(e.toString());
41.         } finally {
42.             try {
43.                 prov.shutdown();    //  Finalizamos el proveedor
44.             } catch (Exception ex){}
45.         }
46.     }
47. }
48.
```

Ejemplo: Descolgado automático de todas las llamadas que suenen en un terminal.

2009-02-26
Introducción teórica a XPath

2009-02-26
Integración Selenium / Maven 2 / Surefire / Cargo / Tomcat 6

2009-02-24
Selenium Remote Control

2009-02-22
Integración de Groovy, JRuby y BeanShell con Spring 2

2009-02-18
Instalación de Pentaho BI Suite Community Edition 1.7.0

2009-02-18
Replicar Web PHP en máquina local

2009-02-16
Selenium Core : El motor de Selenium.

2009-02-16
Integración de JasperReports con PHP

2009-02-09
EJB 3.0 y pruebas unitarias con Maven, JUnit 4 y Embedded JBoss sobre Java 6

2009-02-09
Web Service Security

2009-02-09
Manual Avanzado de Firebug

2009-01-29
Ejemplo con Mockito

2009-01-29
Uso de Mock objects en pruebas con Mockito

2009-01-29
StrutsTestCase

2009-01-28
Eventos en Hibernate (parte III)

2009-01-28

view plain print ?

```
01. import javax.telephony.*;
02. import javax.telephony.TerminalConnection;
03. import javax.telephony.events.CallEv;
04. import javax.telephony.events.TermConnRingingEv;
05.
06. /**
07.  * Durante un tiempo, descuelgará todas las llamadas que suenen en un termina
08.  * @author Carlos García. Autentia
09.  */
10. public class DescolgandoLlamadasApp {
11.     public static void main(String args[]) {
12.         JtapiPeer peer = null;
13.         Provider prov = null;
14.         Terminal term = null;
15.         java.lang.String device = null;
16.         CallObserver myCallObserver = null;
17.
18.         try {
19.             if (args.length != 1) {
20.                 System.out.println("Formato \"DescolgandoLlamadasApp <terminal>\"");
21.                 System.exit(1);
22.             }
23.             // Obtenemos la implementación javax.telephony.JtapiPeer.
24.             peer = JtapiPeerFactory.getJtapiPeer(null);
25.             // Obtenemos un proveedor que de el servicio. Por defecto Core.
26.             prov = peer.getProvider(null);
27.
28.             // Obtenemos el terminal del cual vamos a descolgar las llamadas.
29.             term = prov.getTerminal(args[0]);
30.
31.             myCallObserver = new MiBonitoCallObserver();
32.             term.addCallObserver(myCallObserver);
33.
34.             Thread.sleep(25000); // Descuelga las llamadas durante un tiempo
35.
36.       term.removeCallObserver(myCallObserver);
37.         } catch (Exception ex) {
38.             System.out.println(ex.toString());
39.         } finally {
40.             // Finalizamos el proveedor.
41.             try {
42.                 prov.shutdown();
43.             } catch (Exception ex) {}
44.     }
45.     }
46. }
47.
48. class MiBonitoCallObserver implements CallObserver {
49.   public void callChangedEvent(CallEv[] ev) {
50.         TerminalConnection tc;
51.
52.     for (int i = 0; i < ev.length; i++) {
53.         if (ev[i].getID() != TermConnRingingEv.ID){
54.             continue;
55.         }
56.
57.         try {
58.             tc = ((TermConnRingingEv) ev[i]).getTerminalConnection();
59.             tc.answer();
60.         } catch (Exception ex) {
61.             // No se dará
62.         }
63.     }
64.   }
65. }
```

Paquete CallControl

Este paquete extiende el paquete Core, permitiendo la siguiente funcionalidad:

1. Transferencia de llamadas tanto directas como indirectas.
2. Realizar conferencias.
3. Redireccionar llamadas.
4. Descolgar llamadas que están sonando en otro terminal.
5. Realizar llamadas de consulta.
6. Agregar un terminal a una conferencia ya establecida.
7. Retener y recuperar llamadas.
8. Estados más finos sobre llamadas, terminales, conexiones, direcciones, etc.

Eventos en Hibernate (parte II)

2009-01-27
Eventos en Hibernate (parte I)

2009-01-25
Aprendiendo XMLSchema a través de ejemplos

2009-01-20
Pruebas Software con Junit 4 y Eclipse

2009-01-19
Executor : Un programa para ejecutarlos a todos.

2009-01-18
Soap Monitor: Monitorización de mensajes SOAP en Axis2

2009-01-16
Restaurar una Base de Datos en SQL Server o como cambiar el propietario de los objetos de la base de datos

2009-01-14
Solución a NoClassDefFoundError: SWTResourceUtil

2009-01-14
Desarrollo de aplicaciones Web con Struts 1

2009-01-07
Log4J: Cómo crear un log que trabaje hacia una Base de Datos.

9. Información del número de teléfono de todos los interlocutores de la llamada.

Ejemplo: Realizar una llamada y transferirla a otro puesto.

view plain print ?

```
01. import javax.telephony.*;
02. import javax.telephony.events.*;
03. import javax.telephony.callcontrol.*;
04. import javax.telephony.callcontrol.events.*;
05. 
06. /**
07.  * Ejemplo de transferencias (indirecta) de llamadas
08.  * @author Carlos García. Autentia
09.  */
10. public class LlamarYTransferirApp {
11. 
12.     /**
13.      * Realizamos una llamada del terminal 430 al 777 y esperamos que descuel
14.      * realizamos otra llamada del 430 al 555 y el tranferimos la primera lla
15.      */
16.     public static void main(String[] args) {
17.         JtapiPeer       peer = null;
18.         Provider        prov = null;
19.         CallControlCall call = null;
20.         CallControlCall call2 = null;
21.         Terminal        term = null;
22.         TerminalConnection[] tcs = null;
23.         Connection[]    cons = null;
24. 
25.         try {
26.             peer = JtapiPeerFactory.getJtapiPeer(null);
27.             prov = peer.getProvider(null);
28. 
29.             term = prov.getTerminal("430");
30. 
31.             call = (CallControlCall) prov.createCall();
32.             call.connect(term, term.getAddresses()[0], "777");
33. 
34.             // Esperamos unos segundos en los que debe descolgar.
35.             Thread.sleep(8000);
36. 
37.             call2 = (CallControlCall) prov.createCall();
38. 
39.             tcs = call.getConnections()[0].getTerminalConnections();
40.             cons = call2.consult(tcs[0], "555");
41. 
42.             call.setTransferController(call.getConnections()
    [0].getTerminalConnections()[0]);
43.             call2.setTransferController(call2.getConnections()
    [0].getTerminalConnections()[0]);
44. 
45.             // Realiamos la tranferencia
46.             call.transfer(cons[0].getCall());
47. 
48.         } catch (Exception e) {
49.             System.out.println(e.toString());
50.         } finally {
51.             try {
52.                 // Liberamos recursos
53.                 prov.shutdown();
54.             } catch (Exception ex){}
55.         }
56.     }
57. }
58. 
```

Ejemplo: Conferencia de llamadas.

```
view plain  print  ?
01. import javax.telephony.*;
02. import javax.telephony.events.*;
03. import javax.telephony.callcontrol.*;
04. import javax.telephony.callcontrol.events.*;
05. 
06. /**
07.  * Ejemplo de conferencia de llamadas.
08.  * @author Carlos García. Autentia
09.  */
10. public class EstablecerConferenciaApp {
11.     /**
12.      * Pasos:
13.      *  1) Llamamos del terminal 555 al 666.
14.      *  2) Llamamos del terimnal 555 al 777.
15.      *  3) Realizamos la conferencia.
16.      */
17.     public static void main(String args[]) {
18.         JtapiPeer       peer = null;
19.         Provider        prov = null;
20.         CallControlCall call1 = null;
21.         CallControlCall call2 = null;
22.         Connection[] connections1 = null;
23.         Connection[] connections2 = null;
24.         TerminalConnection[] tcs = null;
25. 
26.         try {
27.             peer = JtapiPeerFactory.getJtapiPeer(null);
28.             prov = peer.getProvider(null);
29. 
30.             call1 = (CallControlCall) prov.createCall();
31. 
32.             connections1 = call1.connect(prov.getTerminal("555"), prov.getAddress("5
33. 
34.             Thread.sleep(2000);
35. 
36.             // Realizamos la segunda llamada
37.             call2 = (CallControlCall) prov.createCall();
38. 
39.             connections2 = call2.connect(prov.getTerminal("555"), prov.getAddress("5
40. 
41.             Thread.sleep(2000);
42. 
43.             // Configuramos los controladores de conferencias
44.             call1.setConferenceController(connections1[0].getTerminalConnections()[0
45.             call2.setConferenceController(connections2[0].getTerminalConnections()[0
46. 
47.             ((CallControlTerminalConnection) call1.getConferenceController()).unhold(
48. 
49.             // Realizamos la conferencia
50.             call1.conference(call2);
51. 
52.             Thread.sleep(2000);
53.         } catch (Exception e) {
54.             System.out.println(e.toString());
55.         } finally {
56.             try {
57.                 // Liberamos recursos
58.                 call1.drop();
59.                 call2.drop();
60.                 prov.shutdown();
61.             }catch (Exception ex){}
62.         }
63.     }
64. }
65. 
```

Paquete CallCenter

Este paquete extiende el paquete Core, permitiendo la siguiente funcionalidad:

1. Operaciones con grupos de agentes (consultar y modificar).
2. Obtener información de llamdas por grupo de agentes (teleoperadores, etc.).
3. Consultar estados de agentes.
4. Asociar datos (objetos) a llamadas.
5. Realizar llamadas predictivas (muy usado en marcadores automáticos).

Ejemplo: Dar de alta a un agente en un grupo.

```
view plain  print  ?
01. import javax.telephony.*;
02. import javax.telephony.callcenter.*;
03.
04. /**
05.  * Damos de alta al teleoperador "carlos_garcia" en el grupo 150, para ello u
06.  * @author Carlos García. Autentia
07.  */
08. public class LoggedOnApp {
09.
10.     public static void main(String[] args){
11.         JtapiPeer           peer = null;
12.         CallCenterProvider  prov = null;
13.         AgentTerminal       term = null;
14.         ACDAddress[]        groups = null;
15.         Agent               agente = null;
16.
17.         try {
18.             peer   = JtapiPeerFactory.getJtapiPeer(null);
19.             prov   = (CallCenterProvider) peer.getProvider(null);
20.             groups = prov.getACDAddresses();
21.             term   = (AgentTerminal) prov.getTerminal("555");
22.             agente = term.addAgent(term.getAddresses()[0], groups[150], Agent.READY,
23.
24.         } catch ( Exception e ){
25.             System.out.println(e);
26.         } finally {
27.             try {
28.                 prov.shutdown();
29.             } catch (Exception ex){}
30.         }
31.     }
32. }
33.
```

Ejemplo: Realizamos una llamada telefónica y le asociamos información de negocio.

```
view plain  print  ?
01. import javax.telephony.*;
02. import javax.telephony.callcenter.*;
03.
04. /**
05.  * Realizamos una llamada y le asociamos datos (cualquier objeto serializable
06.  * @author Carlos García. Autentia
07.  */
08. public class AssingDataToCallApp {
09.
10.     public static void main(String[] args) {
11.         JtapiPeer           peer = null;
12.         CallCenterProvider  prov = null;
13.         CallCenterCall      call = null;
14.         Connection[]        connections = null;
15.
16.         try {
17.             peer = JtapiPeerFactory.getJtapiPeer(null);
18.             prov = (CallCenterProvider) peer.getProvider(null);
19.
20.             call = (CallCenterCall) prov.createCall();
21.
22.             // Realizamos una llamada
23.             connections = call.connect(prov.getTerminal("555"), prov.getAddress("555'
24.
25.             // Asociamos datos a la llamada
26.             call.setApplicationData("En un lugar de la mancha...");
27.
28.         } catch (Exception ex) {
29.             System.out.println(ex);
30.         } finally {
31.             try {
32.                 // Liberamos recursos
33.                 prov.shutdown();
34.             } catch (Exception ex){}
35.         }
36.     }
37. }
38.
```

Paquete Phone

Este paquete extiende el paquete Core, permitiendo la siguiente funcionalidad:

1. Consultar y modificar la pantalla, el volumen y los indicadores luminosos de un terminal.
2. Pulsar los botones del teclado del terminal.
3. Monitorizar los cambios en la pantalla y detectar pulsaciones de las teclas.

Ejemplo: Pulsación de teclas.

view plain print ?

```
01.
02. import javax.telephony.*;
03. import javax.telephony.phone.*;
04.
05. /**
06.  * Pulsamos la tecla 5 dos veces
07.  * @author Carlos García. Autentia
08.  */
09. public class PhoneTerminalPressKeyApp {
10.     public static void main(String[] args){
11.         JtapiPeer peer = null;
12.         Provider  prov = null;
13.         Terminal  term = null;
14.         ComponentGroup[] groups = null;
15.         Component[] botones = null;
16.
17.         try {
18.             peer = JtapiPeerFactory.getJtapiPeer(null);
19.             prov = peer.getProvider(null);
20.             term = prov.getTerminal(null);
21.
22.             groups = ((PhoneTerminal) term).getComponentGroups();
23.             for ( int i = 0; i < groups.length; i++ ){
24.                 if ( "Button".equalsIgnoreCase(groups[i].getDescription())){
25.                     botones = groups[i].getComponents();
26.                     break;
27.                 }
28.             }
29.
30.             PhoneTerminalPressKeyApp.PressKey(botones, "5");
31.             PhoneTerminalPressKeyApp.PressKey(botones, "5");
32.         } catch ( Exception ex ){
33.             System.out.println(ex);
34.         } finally {
35.             try {
36.                 prov.shutdown();
37.             } catch (Exception ex){}
38.         }
39.     }
40.
41.     /**
42.       * Presiona un tecla por su identificador
43.       */
44.     private static void PressKey(Component[] botones, String buttonID){
45.         PhoneButton btn;
46.
47.         for ( int i = 0; i < botones.length; i++){
48.             btn = (PhoneButton) botones[i];
49.             if (btn.getName().equalsIgnoreCase(buttonID) ){
50.                 btn.buttonPress();
51.                 return;
52.             }
53.         }
54.     }
55. }
```

Referencias

Especificación JTAPI

Conclusiones

Comparada con otras APIs de teléfonia, desde mi punto de vista JTAPI gana por goleada en sencillez debido a su diseño orientado a objetos. En su ultima versión (1.4) No le falta de nada, eso sí, cada proveedor de telefonía puede o no implementar lo que dice la especificación, para ello primeramente se debería de preguntar por las capacidades del mismo en relación a cada método de cada clase (tienen métodos getCapabilities()).

Un saludo, espero que os haya resultado de ayuda.

Carlos García.

¿Qué te ha parecido el tutorial? Déjanos saber tu opinión y ¡vota!

Muy malo Malo Regular Bueno Muy bueno

Votar

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.

Nombre: E-Mail:

Comentario:

Enviar comentario **Texto Legal y condiciones de uso**

- Puedes inscribirte en nuestro servicio de notificaciones haciendo clic aquí.
- Puedes firmar en nuestro libro de visitas haciendo clic aquí.
- Puedes asociarte al grupo AdictosAlTrabajo en XING haciendo clic aquí.
- Añadir a favoritos Technorati. ADD THIS BLOG TO MY Technorati FAVORITES

Recuerda

Autentia te regala la mayoría del conocimiento aquí compartido (Ver todos los tutoriales). Somos expertos en: J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ... y muchas otras cosas.

¿Nos vas a tener en cuenta cuando necesites consultoría o formación en tu empresa?, ¿Vas a ser tan generoso con nosotros como lo tratamos de ser con vosotros?

Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos ...

Autentia = Soporte a Desarrollo & Formación.

info@autentia.com

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	Valoración	Votos	Pdf
Creando Servicios Web con Bea Workshop 8.1	En este artículo os mostramos como funciona la nueva herramienta de Bea, Workshop y como crear servicios web con ella	2003-12-15	12174	Muy bueno	2	
SOA y el modelo de Negocio	En este tutorial podemos ver una documento de Santiago Pereira sobre SOA, los estándares que se usan y la unión del SOA con los procesos de negocio.	2007-04-23	3613	Muy bueno	1	
AspectJ, Programación con Aspectos	Os mostramos como configurar AspectJ (extensión Java para la programación basada en aspectos) y un pequeño ejemplo para medir la velocidad de una función sin alterar su código.	2004-01-30	16392	Muy bueno	3	
Persistencia con Spring	En el siquiente tutorial vamos a ver algunas de las aportaciones que nos ofrece Spring para mejorar la capa de persistencia de nuestras aplicaciones	2009-03-04	1169	Muy bueno	9	
Introducción a Lucene	En este tutorial realizaremos una breve introdución a API de búsqueda Lucene	2008-08-17	2085	Muy bueno	7	
Gestor de Contenidos Gratuito con Typo3	César Crespo nos enseña como instalar y utilizar uno de los mejores gestores de contenidos gratuitos del mercado	2004-07-22	19186	Muy bueno	2	
Aplicación profesional con Struts	En este tutorial, os mostramos como crear una aplicación profesional, usando numerosos patrones y utilizando Struts.	2003-10-08	131427	Muy bueno	34	
J2ME. Internacionalización de aplicaciones para móviles	En este tutorial aprenderemos a internacionalizar MIDlets mediante el estandar JSR-238	2008-05-06	2603	Bueno	8	
Servicio Web con NetBeans 6 y prueba con SoapUI	En este tutorial os enseñamos cómo crear y probar un servicio web de una manera sencilla utilizando netbeans 6	2007-08-02	9704	Bueno	6	
CMMI. Modelo de Madurez Software	Os introducimos a CMMI o Capability Maturity Model Integration. CMMI es un modelo de calidad exigido por el gobierno americano a sus proveedores para el desarrollo de Software. Su conocimiento es esencial para reducir costes de desarrollo.	2004-02-05	66243	Bueno	25	

Nota:

Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento. Los contenidos y comentarios de los tutoriales son responsabilidad de sus respectivos autores. En algún caso se puede hacer