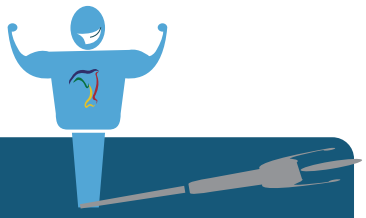
autentia

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...

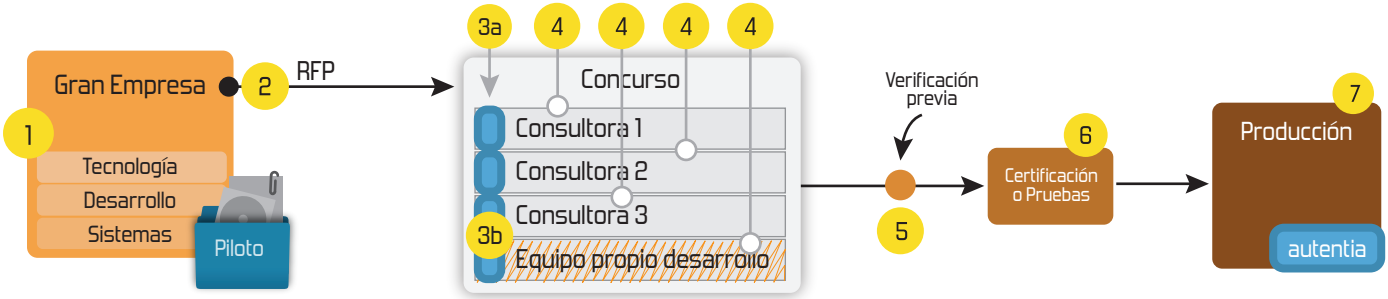
1. Desarrollo de componentes y proyectos a medida



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: Cesar Crespo Martín 2003.

Si te gusta lo que ves, **puedes contratarnos** para impartir **cursos presenciales** en tu empresa o para ayudarte en proyectos (Madrid).

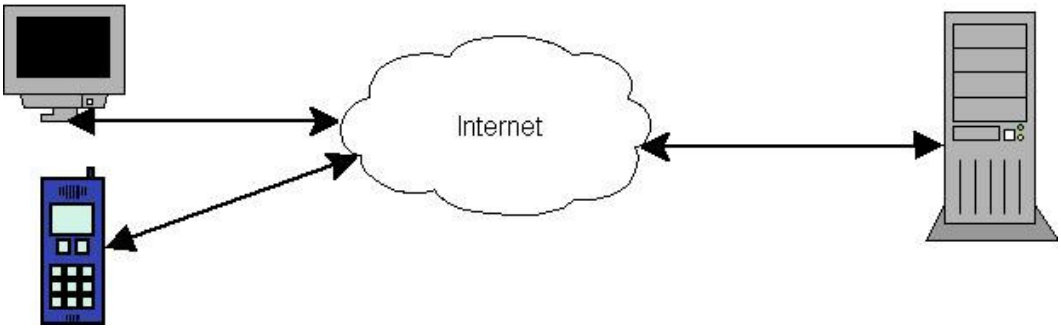
Contacta: cesar@autentia.com.

Descargar este documento en formato PDF soap.pdf

Introducción a los Web Services con PocketSOAP, Apache SOAP y Axis:

Los sistemas distribuidos y el acceso a servicios remotos han centrado gran parte de los esfuerzos tecnológicos de los últimos años habiéndose creado distintas tecnologías y estándares distintos como por ejemplo: Java RMI, CORBA, COM/DCOM, las clásicas RPCs o los sistemas MOM basados en el paso de mensajes.

Con el crecimiento de la WWW surgió la idea de realizar estas tareas en sistemas distribuidos usando un protocolo estandarizado como HTTP:



Los Web Services son básicamente funciones que se ejecutan en un servidor y son accedidas por los clientes mediante Internet.

Para realizar este tipo de comunicación necesitamos definir un formato en el que se transmitirán esos datos. Aquí es donde entre en juego SOAP, WSDL y UDDI:

Web Services = SOAP + UDDI + WSDL

SOAP , WSDL y **UDDI**:

Simple Object Access Protocol (SOAP) (a veces también referido como Service Oriented Access Protocol) actualmente es un protocolo de especificación que define una forma uniforme de transmitir datos codificados en XML. A su vez también define una forma de realizar llamadas a procedimientos remotos (similar a RPCs).

La definición oficial que nos da el consorcio W3C: es la siguiente:

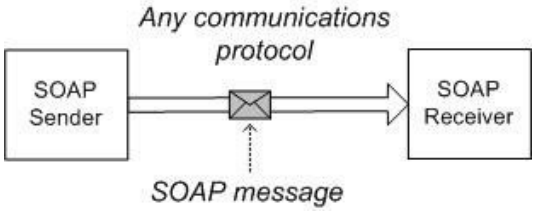
> *SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. SOAP uses XML technologies to define an extensible messaging framework, which provides a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics.*

SOAP 1.1	
Namespace Name	http://schemas.xmlsoap.org/soap/envelope/
Spec Location	http://www.w3.org/TR/SOAP/
SOAP 1.2	
Namespace Name	http://www.w3.org/2002/12/soap-envelope
Spec Location	http://www.w3.org/TR/soap12-part0/ (Primer) http://www.w3.org/TR/soap12-part1/ http://www.w3.org/TR/soap12-part2/

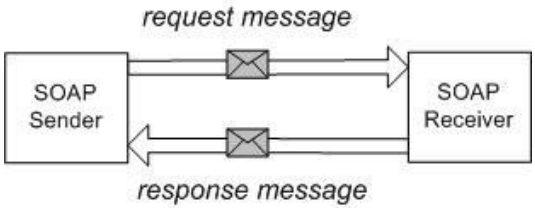
Originalmente los autores de SOAP confiesan que se centraron en el "acceso a objetos" de ahí la O de las siglas, pero con el tiempo SOAP se centró en una forma de RPCs vía web.

SOAP define el modo de transmitir mensajes XML de un punto a otro. Lo hace proporcionando un protocolo de mensajes que es:

1. Extensible (se basa en XML).
2. Usable por una gran variedad de protocolos de transporte de red. Una gran ventaja de SOAP es que puede usar cualquier protocolo de transporte como TCP, HTTP, SMTP o incluso MQs (colas de mensajes)
3. Independencia de las plataformas o lenguajes de programación



SOAP engloba cada mensaje dentro de un *sobre* (envelope):



Por ejemplo supongamos que queremos acceder a una función llamada getQuote al que le pasamos el nombre de una empresa y nos devuelve un *float* con el valor su *Stock Quote.*

Si se tratara de una función normal por ejemplo realizaríamos esta llamada float q = getQuote("XXX"); devolviéndonos el valor 97.8.

Si realizáramos esta operación pidiendo el resultado a un Web Service los datos enviados y recibidos serían los siguientes.

Mensaje de Petición:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
xmlns:n="urn:xmethods-delayed-quotes"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<n:getQuote>
<symbol xsi:type="xs:string">XXX</symbol>
</n:getQuote>
</soap:Body>
</soap:Envelope>
```

Mensaje de Respuesta:

```
<soap:Envelope soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
  <soap:Body>
    <n:getQuoteResponse xmlns:n="urn:xmethods-delayed-quotes">
      <Result xsi:type="xsd:float">97.8</Result>
    </n:getQuoteResponse>
  </soap:Body>
</soap:Envelope>
```

Observamos que el formato del mensaje consiste en un *Envelope* donde se incluyen datos en XML. También observamos que se definen tipos predefinidos: **xsd:float**

La codificación del envelope puede estar codificado según diferentes filosofías:

- rpc encoded:
  Aquí se codifcan los nuestros datos siguiendo el schema http://schemas.xmlsoap.org/soap/envelope/

- doc /literal xml
  Esta filosofía es la elegida por .NET donde el contenido del envelope se codifica como XML literal.

- mensaje
  Filosofía en la que el contenido del cuerpo del envelope es un mensaje con el formato XML

Para no tener que generar manualmente estos mensajes XML diferentes implementaciones de Web Services proporcionan APIs y frameworks tanto para programar clientes como Web Services en distintos lenguajes.

Aquí podemos encontrar una lista con algunas implementaciones para consumir y producir Web Services:

http://www.xmethods.com/ve2/ViewImplementations.po

Como podemos observar existen varias posibles implementaciones dependiendo de la plataforma.

En http://www.xmethods.com también podemos encontrar una serie de Web Services consumibles por los clientes que programemos para comprobar su funcionamiento y su interoperabilidad.

Una de las grandes posibilidades de los Web Services es proporcionar una gran interoperabilidad entre diferentes clientes y servicios. Para ello necesitamos que el cliente sea capaz de obtener una descripción del servicio con el que se va a comunicar. Para esto existe el **W**eb **S**ervice **D**efinition **L**anguage. **WSDL** es un formato que describe las funciones de un web service así como los parámetros de entrada y salida que éste tiene. Se podría decir que WSDL es el IDL (Interface Definition Language) de los Web Services

Si además queremos proporcionar transparencia de localización a nuestro servicio web usaremos **UDDI**, Universal Description, Discovery and Integration protocol. UDDI es un protocolo que crea una plataforma estándar e interoperable que permite a las compañías y aplicaciones, encontrar servicios dinámicamente en Internet (http://www.uddi.org/.).

Nosotros usaremos como cliente una PDA con Windows Pocket PC 2003 usando Embedded Visual C++ 4.0. Elegimos la implementación PocketSoap (http://www.pocketsoap.com/pocketsoap/) para la parte cliente y para crear Web Services Axis (http://ws.apache.org/axis/) aunque también explicaremos Apache SOAP (http://ws.apache.org/soap/). Los Web Services correrán sobre Tomcat.

Para el desarrollo del cliente SOAP en para Windows Pocket PC 2003 usaremos el entorno Embedded Visual C++ (eVC++) y el emulador de Pocket PC 2003.

PocketSOAP:

PocketSOAP es un componente COM cliente Open Source para la familia Windows , que se proporciona con la MPL (Mozilla Public License). Originalmente se diseñó para PocketPC, pero también existe una versión que funciona sobre Windows 95/98/Me/NT4/2000/XP:

http://www.pocketsoap.com/pocketsoap/

Este objeto COM puede ser usado desde nuestras aplicaciones escritas en distintos lenguajes, como por ejemplo Visual Basic y Visual C++.

Instalación de las librerías de PocketSoap en el emulador Pocket PC 2003

Para instalar las librerías de PocketSoap en nuestro emulador debemos bajarnos el paquete para el emulador de:

http://www.pocketsoap.com/pocketsoap/

Nosotros usaremos esta versión:

binaries for PocketPC 2002 emulator:http://www.pocketsoap.com/pocketsoap/pocketsoap_pp2e_1.4.3.zip

A pesar de indicar que son para el emulador Pocket PC 2002 también funcionan para el emulador Pocket PC 2003, que será el que usaremos.

PocketSoap lo componen una serie de controles ActiveX. Estos controles tienen varios objetos COM que ofrecen interfaces para comunicarnos con Web Services.

Para que nuestras aplicaciones puedan usar estos controles ActiveX y puedan instanciarlos como objeto COM debemos copiar las DLLs y luego registrarlas en el sistema operativo. En el caso de Win32 usamos el programa regsvr32.exe, y en el caso de Pocket Windows 2003 usamos regsvrce.exe.

En el caso de Win32 la aplicación regsvr32.exe se encuentra incluida con Winodws. En cambio tanto en el emulador como en nuestra pda con Pocket PC 2003 la aplicación regsvrce.exe no viene incluida. Esta aplicación la encontramos dentro del Microsoft Embedded C++ 4.0:

Para el caso del emulador:

C:\bin\MicrosofteMbeddedC++4.0\EVC\WCE400\TARGET\X86

Para el caso de pdas con procesadores ARM y X-Scale

C:\bin\MicrosofteMbeddedC++4.0\EVC\WCE400\TARGET\ARMV4

Para poder usar desde nuestro programa las DLLs proporcionadas por PocketSoap tenemos que copiarlas al directorio \Windows y registrarlas con regsvrce.exe.

Para realizar esta tarea usamos la capacidad del emulador para compartir directorios de nuestro PC. Este directorio se mostrará dentro del emulador como una tarjeta de memoria. Por lo tanto copiamos las DLLS:

- PocketHTTP.dll
- psDime.dll

- pSOAP.dll
- xmlparse.dll
- xmltok.dll
- zlib.dll

Al directorio que compartamos:

Abrimos el explorador de ficheros (Inicio>Programas>Explorador de ficheros) y nos dirigimos a la "Tarjeta de Almacenamiento" donde se nos mostraran los ficheros del directorio del pc que compartimos.

Observamos que las DLLS no aparecen, por lo que deberemos seleccionar la opción "Ver todos los ficheros":


Pocket PC 2003
Emulator Help
Pocket PC
File Explorer
1:39
Storage Card
Name
CEMON 3/17/03 159K
JitDebugger 3/17/03 16.5K
REGSVRCE
WebServices ...
Refresh
View All Files
Paste
Paste Shortcut
New Folder
Edit Open

Copiamos los ficheros y los pegamos en \Windows

Windows Pocket PC 2003 no tiene una Consola de comandos desde la cual poder ejecutar programas y pasarle parámetros. En cambio si tiene un menú para ejecutar comandos. Para que aparezca este menú deberemos situarnos encima del reloj con el cursor y hacer click mientras pulsamos la tecla Control hasta que aparezca el siguiente menú:

En el caso de una PDA real en vez de pulsar la tecla control pulsaremos el botón principal de la PDA y obtendremos el mismo resultado:

En estos momentos tenemos las DLLs que necesitamos en el directorio \Windows así como el programa regsvrce.exe para registrarlas en el sistema.

En caso de no registrar correctamente las DLLs nuestro programa producirá el siguiente error:

Para que nuestros programas puedan usar PocketSOAP correctamente debemos ejecutar regsvrce con las siguientes DLLs:

- pocketHTTP.dll
- psoap.dll
- psdime.dll


Pocket PC 2003
Emulator Help
Pocket PC
Run
1:36
Thursday, February 19, 2004
Run
Type the name of a program, folder, or document, and Windows will open it for you.
Open:
regsvrce psoap.dll
OK
Cancel
New

Desgraciadamente esta tarea tendremos que realizarla cada vez que apaguemos y reiniciemos el emulador.

Instalación de las librerías de PocketSoap en la PDA (iPaq 2210)

Para instalar las librerías de PocketSoap en nuestra PDA debemos bajarnos el paquete para Pocket PC de:

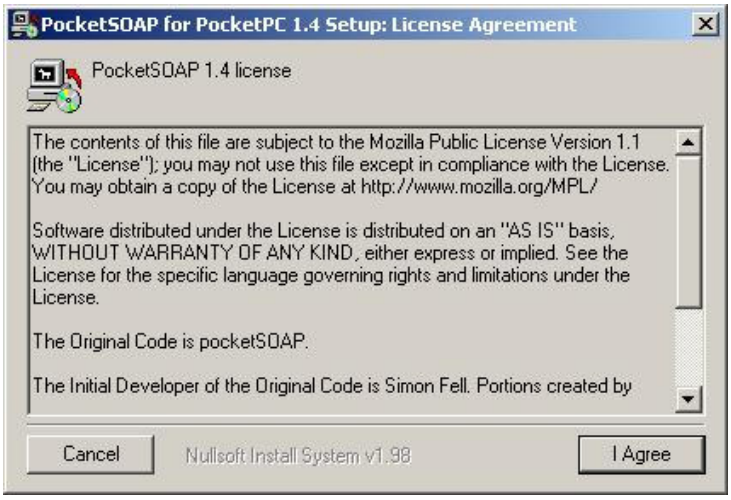
http://www.pocketsoap.com/pocketsoap/

Nosotros usaremos esta versión:

PocketPC Version v1.4.3 Packaged Install

que es compatible con procesadores ARM y X-Scale como nuestra iPaq 2210:

Para instalar correctamente tendremos que tener nuestra PDA conectada al PC y ejecutar desde el PC el programa de instalación:



Este instalador creado con NSIS copiara un ejemplo ejecutable, las dlls correspondientes y las registrará en el sistema Windows Pocket PC 2003 por nosotros.

Ejemplo de cliente SOAP con PocketSoap:

En la página de PocketSOAP existe una sección de ejemplos de clientes SOAP: Samples

En la sección de eVC encontramos el ejemplo getQuote:

PocketPC Stock Quote

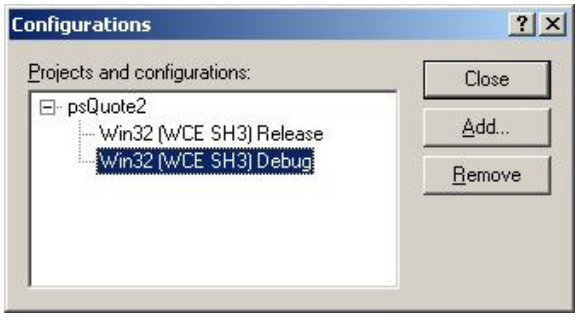
This is the eVC source code for the Stock Quote demo that installs with PocketSOAP on PocketPC's. This is calling the XMethods stock quote service.

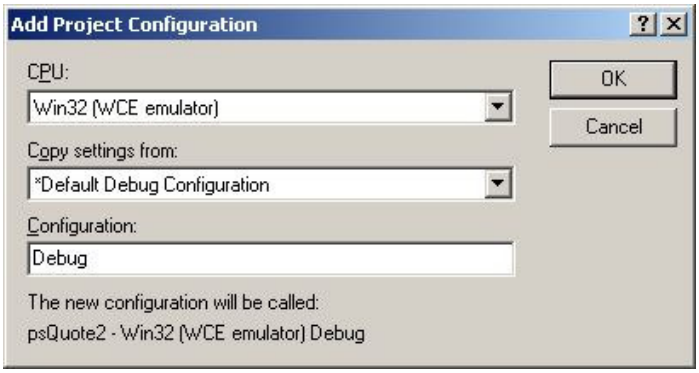
Como nos indica en esta página este programa se conecta con un web service de la página de XMethods.

Una vez descargado el ejemplo y descomprimido debemos asegurarnos que los ficheros tienen permisos de escritura.

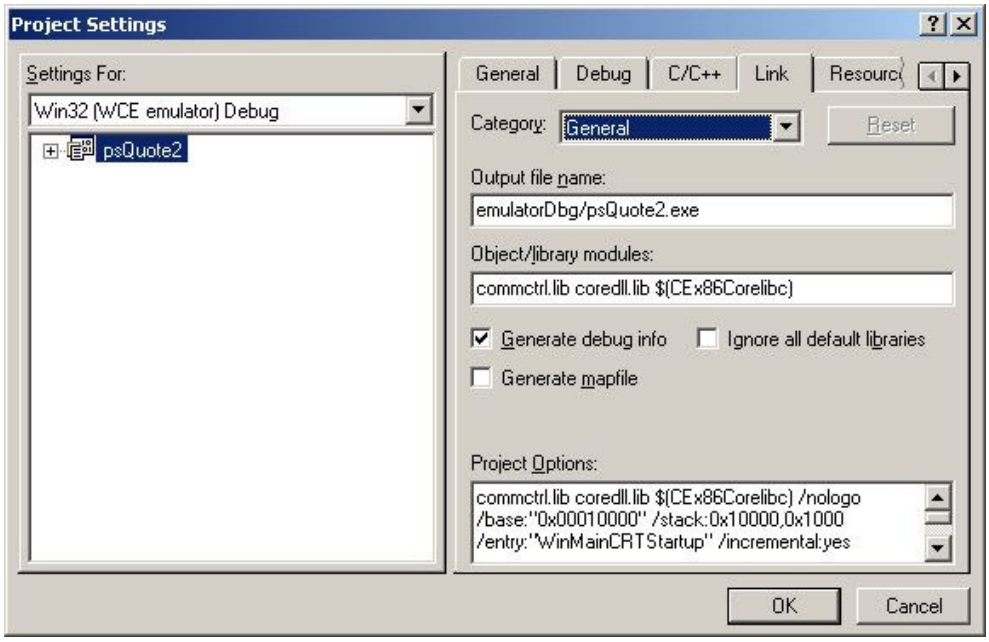
Nuestro entorno de desarrollo será Embedded Visual C++ 4.0 con el emulador de Pocket PC 2003. para que el ejemplo funcione con este entorno debemos realizar varios cambios:

- pSOAP.h: El ejemplo getQuote en el fichero psQuote2Dlg.cpp hace referencia al fichero pSOAP.h, este fichero se encuentra dentro del código fuente de PocketSOAP: v1.4.3 Source Code, por lo que tendremos que añadir este fichero a nuestro proyecto e incluirlo: #include "pSOAP.h".
- Para poder usar este ejemplo con el emulador de Pocket PC 2003 debemos crear una nueva configuración. Para ello seleccionamos en el menú Build > Configurations





- Finalmente en el debemos seleccionar en el menú Project > Settings y en la pestaña Link:



Debemos cambiar /entry: "WinMainCRTStartup" por /entry: "**w**WinMainCRTStartup"

Finalmente tendremos el ejemplo de PocketSOAP funcionando:

En http://www.xmethods.com podemos encontrar varios servicios web con los que podemos probar nuestro cliente.

Para dudas del API de PocketSoap, solución de dudas o errores en Yahoo esta el siguiente grupo que sin duda nos será de gran ayuda:

http://groups.yahoo.com/group/pocketsoap/

Apache SOAP:

Apache Soap es una antigua implementación de SOAP para la creación de Web Services. Actualmente se ha visto superada por Axis y no se prevé que implemente nuevas funcionalidades, cosa que sí hará Axis. Pese a todo por ser una de las primeras implementaciones de SOAP para la creación de Web Services la mencionamos:

Instalación de Apache SOAP

Para poder crear un web service necesitamos descargarnos Apache SOAP de http://ws.apache.org/soap/

Una vez descargado para crear nuestro web service necesitamos copiar las librerías de soap en el directorio /lib de nuestra aplicación.

Para hacer deploy de nuestro servicio ejecutamos el siguiente programa:

java org.apache.soap.server.ServiceManagerClient http://localhost:8080/soap/servlet/rpcrouter deploy DeploymentDescriptor.xml

Usando el descriptor:

```
<isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment"
             id="urn:adictos-delayed-quotes">
 <isd:provider type="java"
               scope="Application"
               methods="getQuote">
   <isd:java class="samples.stockquote.StockQuoteService"/>
 </isd:provider>
 <isd:faultListener>org.apache.soap.server.DOMFaultListener</isd:faultListener>
</isd:service>
```

Siendo samples.stockquote.StockQuoteService la siguiente clase (debe ser compilada previamente):

```
public class StockQuoteService {
public float getQuote (String symbol) throws Exception {
  return (float)120;
 }
}
```

Para llamar a este web service usaremos en nuestras funciones de llamada de nuestro API la siguiente URL

http://localhost:8080/soap/servlet/rpcrouter

Actualmente Apache SOAP 2.x ha sido sobrepasado por el proyecto AXIS, aunque hacemos mención a Apache SOAP 2.x por ser una de las primeras implementaciones para Java junto con GLUE y otros.

Cliente PocketSOAP y servicio SOAP:

Con nuestro cliente pocketsoap llamamos al web service escrito en Java.

Basándonos en el ejemplo de PocketSOAP:

```
HRESULT hr = CoInitializeEx(0, COINIT_MULTITHREADED ) ;
CComPtr<ISOAPEnvelope> penv ;

_HR( CoCreateInstance(__uuidof(CoEnvelope), NULL, CLSCTX_INPROC, __uuidof(ISOAPEnvelope),
(void **)&penv ) );

_HR( penv->put_MethodName ( CComBSTR(OLESTR("getQuote")) ) );
_HR( penv->put_URI ( CComBSTR(OLESTR("urn:adictos-delayed-quotes")) ) );

CEdit * tkr = (CEdit *)GetDlgItem( IDC_TICKER ) ;
CString strTkr ;
tkr->GetWindowText(strTkr) ;
CComPtr<ISOAPNodes> params ;
_HR( penv->get_Parameters(&params) );
_HR( params->Create ( CComBSTR(OLESTR("symbol")), CComVariant(strTkr), NULL, NULL, NULL, NULL ) ) ;
//Al serializar el envelope en req vemos el XML que vamos a enviar
CComBSTR req ;
_HR( penv->Serialize(&req) );
CComPtr<IHTTPTransportAdv> pt ;
_HR( pt.CoCreateInstance( __uuidof(HTTPTransport) ) );
_HR( pt->put_SOAPAction( CComBSTR(OLESTR("")) ) );
//Esto llama al WS de Apache Soap
_HR( pt->Send ( CComBSTR(OLESTR("http://piccolo:8080/soap/servlet/rpcrouter")), req ) );
```

AXIS: Apache eXtensible Interaction System:

Apache Axis es una implementación de SOAP. Axis es la continuación del proyecto Apache SOAP.

Como se nos indica en su FAQ: http://nagoya.apache.org/wiki/apachewiki.cgi?AxisProjectPages/StandardsSupported Axis implementa los estándares JAX.RPC, JAXM, y SAAJ (interfaces de javax.xml.soap) además de cumplir con el estándar WSDL. Estándares que no cumple su predecesor Apache SOAP 2.x, es por esto por lo que no se llamo Apache SOAP 3.0 y se llamó **AXIS**: **A**pache e**X**tensible **I**nteraction **S**ystem.

> **Q: Which industry standards does Axis v1.1 support?**
>
> A: Axis supports the following specifications:
>
> W3C SOAP [v1.1] and [v1.2 Candidate Recommendation]
> [W3C Web Service Description Language (WSDL) v1.1]
> [Sun SOAP with Attachments API for Java (SAAJ) v1.1]
> [Sun Java API for XML-Based RPC (JAX-RPC) v1.0]
>
> Para profundizar en las diferencias http://nagoya.apache.org/wiki/apachewiki.cgi?AxisProjectPages/Compare:

Apache SOAP	Axis
really old	third generation
really slow	much faster, but not as fast as many
no WSDL support	WSDL support
proprietary API	JAX-RPC API
RPC/encoded only	RPC/encoded and Doc/literal
interoperability issues	very interoperable
extensiblity issues	very extensible
low level API for headers	easy handler support for headers

Desde la página de Axis no se recomienda el uso del antiguo Apache SOAP en favor de Axis.

Instalación de Axis

Para poder ejecutar Web Services con Axis usamos un servidor web como Tomcat.

Descargamos Axis de http://ws.apache.org/axis/releases.htm

Copiamos las librerías de Axis del zip que nos hemos bajado a nuestro directorio lib

Configuramos el fichero web.xml de nuestra aplicación web:

web.xml:

```
<web-app>
<display-name>Apache-Axis</display-name>
<servlet>
<servlet-name>AxisServlet</servlet-name>
<display-name>Apache-Axis Servlet</display-name>
<servlet-class>
org.apache.axis.transport.http.AxisServlet
</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>AxisServlet</servlet-name>
<url-pattern>*.jws</url-pattern>
</servlet-mapping>

<mime-mapping>
<extension>wsdl</extension>
<mime-type>text/xml</mime-type>
</mime-mapping>


<mime-mapping>
<extension>xsd</extension>
<mime-type>text/xml</mime-type>
</mime-mapping>

<welcome-file-list id="WelcomeFileList">
<welcome-file>index.html</welcome-file>
<welcome-file>index.jsp</welcome-file>
<welcome-file>index.jws</welcome-file>
</welcome-file-list>
```

De esta forma cada vez que se nos pida un fichero .jws el servlet AxisServlet será el encargado de saber a lo que hacer con nuestro Web Service.

Para crear nuestro Web Service StockQuote simplemente tenemos que copiar nuestro fichero java al directorio de nuestra aplicación y renombrarlo de .java a .jws (nunca un deploy fue tan fácil, como veremos este tipo de deploy será limitado):

StockQuoteService.jws

```
public class StockQuoteService {
public float getQuote (String symbol) throws Exception {

  return (float)120;
 }
}
```

La funcionalidad de nuestro web service es muy simple ya que solo lo creamos para ver la interacción con otros clientes.

La primera vez que se pida este servicio será compilado automáticamente.

Además podemos ver su descripción en WSDL la cual es generada automáticamente:

```
http://localhost:8080/axis/StockQuoteService.jws?wsdl<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://localhost:8080/axis/StockQuoteService.jws"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://localhost:8080/axis/StockQuoteService.jws"
xmlns:intf="http://localhost:8080/axis/StockQuoteService.jws"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">


 <wsdl:message name="getQuoteRequest">
            <wsdl:part name="symbol" type="xsd:string"/>
 </wsdl:message>
 <wsdl:message name="getQuoteResponse">
  <wsdl:part name="getQuoteReturn" type="xsd:float"/>
 </wsdl:message>
 <wsdl:portType name="StockQuoteService">
  <wsdl:operation name="getQuote" parameterOrder="symbol">
   <wsdl:input message="impl:getQuoteRequest" name="getQuoteRequest"/>
   <wsdl:output message="impl:getQuoteResponse" name="getQuoteResponse"/>
  </wsdl:operation>
 </wsdl:portType>
 <wsdl:binding name="StockQuoteServiceSoapBinding" type="impl:StockQuoteService">
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="getQuote">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="getQuoteRequest">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                                   namespace="http://DefaultNamespace" use="encoded"/>
   </wsdl:input>
   <wsdl:output name="getQuoteResponse">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
              namespace="http://localhost:8080/axis/StockQuoteService.jws"
              use="encoded"/>
</wsdl:output>
  </wsdl:operation>
 </wsdl:binding>

 <wsdl:service name="StockQuoteServiceService">
  <wsdl:port binding="impl:StockQuoteServiceSoapBinding" name="StockQuoteService">
   <wsdlsoap:address location="http://localhost:8080/axis/StockQuoteService.jws"/>
  </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```

Con nuestro navegador podemos realizar llamadas a los Web Services y comprobar los mensajes SOAP que nos devuelven. Así si llamamos a nuestro servicio StockQuoteService, le podemos indicar el método que queremos llamar con sus parámetros de la siguiente forma:

http://localhost:8080/axis/StockQuoteService.jws?method=getQuote&symbol=XXX

Obteniendo el siguiente resultado:

```
<?xml version="1.0" encoding="UTF-8" ?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"

            xmlns:xsd="http://www.w3.org/2001/XMLSchema"

            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
     <getQuoteResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
                    <getQuoteReturn xsi:type="xsd:float">120.0</getQuoteReturn>
      </getQuoteResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Uso de tipos complejos con Axis

Para tener un mayor control sobre nuestro web service necesitamos un fichero de descripción. Para ello se usa WSDD (Web Service Deployment Descriptor) cuando necesitamos recibir o devolver tipos complejos de datos necesitamos un descriptor de Web service.

Supongamos que nuestro web service quiere devolver un objeto del tipo Product o un array de Product:

```
package   adictos;

public class Product {
    private java.lang.String id;
    private java.lang.String nombre;
              public Product () {
              }

             (.................)

   }
```

Product debe ser una clase conforme con la especificación de Java Bean, es decir, tener un constructor por defecto, y todos sus métodos get/set.

Supongamos una de las funciones de nuestro Web Service es la siguiente:

```
public Product []getProducts() {
```

Esta función del web service simplemente devuelve un array de productos, internamente puede obtener los datos de una base de datos por una EJB o incluso de otro Web Service.

El descriptor sería el siguiente:

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
<service name="ipaq" provider="java:RPC" >


 <namespace>http://client.axis.ipaq</namespace>
 <parameter name="className" value="ipaq.axis.ws.IpaqWS"/>
 <parameter name="allowedMethods" value="*"/>
 <parameter name="scope" value="Session"/>
<operation name="getProducts" qname="operNS:syncProducts"
                xmlns:operNS="http://client.axis.ipaq"
   </operation>
<typeMapping
             xmlns:ns="http://localhost:8080/axis/services/Product"
             qname="ns:ArrayOf_tns1_Product"
             type="java:ipaq.Product[]"
             serializer="org.apache.axis.encoding.ser.ArraySerializerFactory"
             deserializer="org.apache.axis.encoding.ser.ArrayDeserializerFactory"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"                />


<typeMapping
             xmlns:ns="http://client.axis.ipaq"
             qname="ns:Product"
             type="java:ipaq.Product"
             serializer="org.apache.axis.encoding.ser.BeanSerializerFactory"
             deserializer="org.apache.axis.encoding.ser.BeanDeserializerFactory"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
 </service>
</deployment>
```

En este descriptor se indican los valores de retorno de la función y como serializar este objeto para ser enviado dentro de un soap envelope.

Un serializador es una clase que se encarga de transformar nuestro tipo de datos en xml para incluirlo dentro del soap envelope.

Axis contiene numerosos serializadores, nosotros hemos usado BeanSerializerFactory y ArraySerializerFactory

También podemos crear nuestro propio serializador.

Para hacer el despliegue (deploy) de nuestro web service necesitamos realizar una tarea adicional:

Si usamos ant, bastará con añadir el siguiente target:

```
<target name="deploy" depends="undeploy">

     <java  classname="org.apache.axis.client.AdminClient"   >
          <arg value="${home}/deploy.wsdd"/>
     </java>
</target>
```

```
<target name="undeploy">
      <java  classname="org.apache.axis.client.AdminClient"   >
            <arg value="${home}/deploy.wsdd"/>
      </java>
</target>
```

O ejecutar sobre la línea de comandos el anterior comando:

```
java org.apache.axis.client.AdminClient deploy.wsdd

java org.apache.axis.client.AdminClient undeploy.wsdd
```

Asumiendo en ambos casos que las librerías de Axis se encuentran en el CLASSPATH.

Cliente PocketSOAP y servicio AXIS:

Cuando nuestro cliente usa el servicio AXIS comprobamos que su llamada es mas simple ya que únicamente debemos indicar el método a invocar, copiar los parámetros e indicar donde se encuentra el web service:

```
HRESULT    hr = CoInitializeEx(0,  COINIT_MULTITHREADED ) ;


CComPtr penv ;
_HR( CoCreateInstance(__uuidof(CoEnvelope), NULL, CLSCTX_INPROC,
            __uuidof(ISOAPEnvelope), (void **)&penv ) );

_HR( penv->put_MethodName ( CComBSTR(OLESTR("getQuote")) ) );


CEdit * tkr = (CEdit *)GetDlgItem( IDC_TICKER ) ;
CString strTkr ;
tkr->GetWindowText(strTkr) ;

CComPtr params ;
_HR( penv->get_Parameters(¶ms) );
_HR( params->Create ( CComBSTR(OLESTR("symbol")), CComVariant(strTkr), NULL, NULL, NULL, NULL ) ) ;

//Al serializar el envelope en req vemos el XML que vamos a enviar
CComBSTR req ;
_HR( penv->Serialize(&req) );

CComPtr pt ;
_HR( pt.CoCreateInstance( __uuidof(HTTPTransport) ) );

_HR( pt->Send ( CComBSTR(OLESTR("http://piccolo:8080/axis/StockQuoteService.jws")), req ) );
```

Sin tener que indicar los campos SoapAction ni URI.

Para el diseño de clientes con PocketSOAP actualmente existe un generador de clientes a partir de WSDL en:

http://www.pocketsoap.com/wsdl/

Actualmente solo se encuentra el generador para clientes en Visual Basic.

Cliente PocketSOAP con tipos complejos

Para recibir respuestas de tipos complejos de un web service podemos usar el generador de clientes a partir de WSDL en eVB y traducirlo a eVC++, escribiendo los serializadores y deserializadores correspodientes.

En nuestro caso recorremos la respuesta de nuestro web service que nos devuelve un array de Products de la siguiente forma:

```
_HR( pt->Send ( bstr_url, req ) );
req.Empty() ;
_HR( penv->Parse(CComVariant(pt), NULL) ) ;
params.Release() ;
_HR( penv->get_Parameters(¶ms) ) ;
CComPtr prm ;
_HR( params->get_Item(0, &prm) ) ;
CComVariant resVal ;
_HR( prm->get_Value(&resVal) ) ;


long ub, lb ;
//Recibimos un array de objetos VARIANT
//SI ES VT_NULL el ws nos ha enviado null
if ( resVal.vt !=  (VT_ARRAY | VT_VARIANT)  )
            return NULL;

SafeArrayGetUBound(resVal.parray, 1, &ub);
SafeArrayGetLBound(resVal.parray, 1, &lb) ;
```

```
		CComVariant val;
		CObArray *array  = new CObArray();
		CRegistro *reg = NULL;

#ifdef _DEBUG
		WriteEnvelope(penv,TEXT("DEBUG.xml"));
#endif

		for ( long i = lb ; i <= ub ; ++i ){
			int kk = SafeArrayGetElement(resVal.parray, &i, (void *) & val);
			if(val.vt == VT_UNKNOWN || val.vt == VT_DISPATCH ){
				CComPtr n ;//poner node local aqui
				val.punkVal->QueryInterface(__uuidof(n), (void **)&n) ;

				CComVariant nVal;
				n->get_Value(&nVal);

				if (( nVal.vt == VT_UNKNOWN ) || ( nVal.vt == VT_DISPATCH )){
					CComPtr nd ;
					nVal.punkVal->QueryInterface(__uuidof(nd),(void **)&nd) ;
					CComPtr ct ;
					nd->get_Nodes(&ct) ;
					CComVariant nombre,id;
					CComPtr nodoNombre ;
					ct->get_ItemByName(CComBSTR(L"nombre"), CComBSTR(L""),&nodoNombre ) ;
					nodoNombre->get_Value(&nombre) ;
					CComPtr nodoTipo ;
					ct->get_ItemByName(CComBSTR(L"id"), CComBSTR(L""),&nodoID ) ;
					nodoID->get_Value(&id) ;

					reg = new CRegistro();
					reg->id = id.bstrVal;
					reg->nombre = nombre.bstrVal;

					array->Add(reg);

				}//if
			}//if

		}//for
		SafeArrayUnaccessData(resVal.parray) ;
```

En ambos clientes anteriormente explicados, un parámetro crucial es el *timeout* . Debemos indicar cuanto tiempo ha de esperar nuestro cliente la respuesta del servicio para considerar la ausencia de respuesta como error. Este parámetro deberá ser estudiado en cada caso e indicar el valor que nos convenga de la siguiente forma:

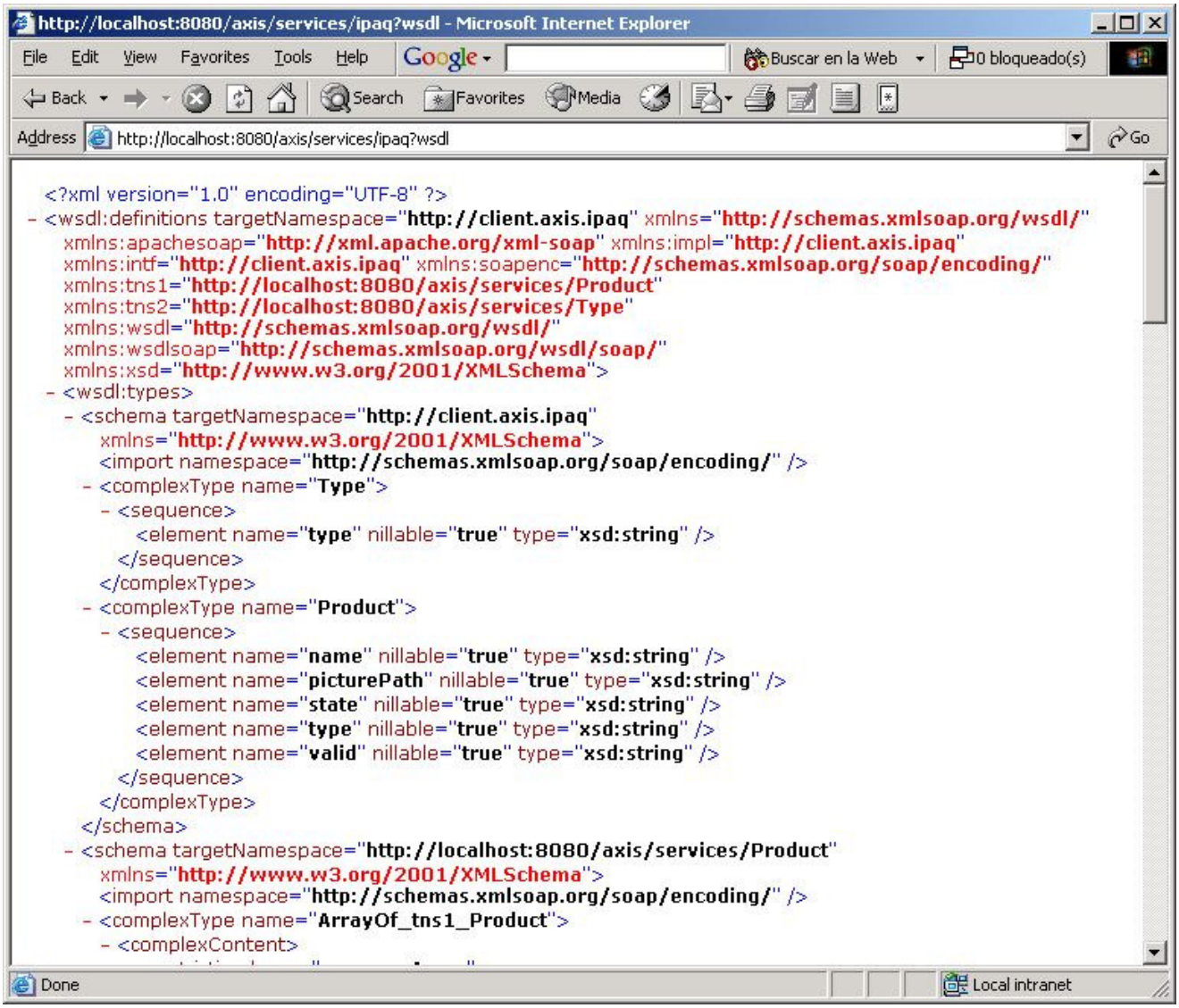
```
pt->put_Timeout(timeout)
```

Cliente en Java

En el caso de Axis la generación del cliente es muy simple.

Para generar un cliente automáticamente, únicamente debemos tener la descripción del web service, es decir su fichero WSDL.

Una forma de obtenerlo es mediante navegador web:



Una vez guardado el fichero ejecutando WSDL2Java obtendremos el cliente con sus Stubs correspondientes así como los casos de test JUnit:

```
java  org.apache.axis.wsdl.WSDL2Java product.wsdl
```

Esta tarea también puede ser automatizada usando Ant:

```
<wsdl2java url="${home}/ipaq.wsdl"
output="${build.dir}/work"
deployscope="session"
serverSide="yes"
skeletonDeploy="yes"
noimports="no"
verbose="no"
typeMappingVersion="1.1"
testcase="yes">
</wsdl2java>
```

Nuevo servicio de notificaciones

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales, inserta tu dirección de correo en el siguiente formulario.

Subscribirse a Novedades	
e-mail	
	Enviar

Otros Tutoriales Recomendados (También ver todos)

Nombre Corto	Descripción
Primeras aplicaciones con Bea Weblogic Platform	Os mostramos como instalar Bea Weblogic Platform así como a crear la primera aplicación, con su entorno visual, utilizando la implementación particular basada en Struts....
SAX, DOM y NetBeans	En este tutorial os vamos a mostrar como manipular documentos XML desde Java y NetBeans
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
Almacenamiento en Windows Pocket 2003	Cesar Crespo nos enseña como utilizar ObjectStore en nuestros programas Visual C++. El "Object Store" (ObS) en Windows Pocket 2003 cumple en muchos casos la misma función que el disco duro en un equipo de escritorio.
Generador automático de Webservices	Os mostramos como crear un servicio Web a partir de una clases, gracias a generadores automáticos de código y NetBeans
AdRotator en Página ASP .Net	Ismael Caballero nos cuenta como utilizar el control AdRotator en una página ASP .Net
Desarrollo Struts con XDoclet	Alejandro Perez nos enseña como simplificar el desarrollo de aplicaciones J2EE basadas en Struts, automatizando la generación de código con XDoclet
Desarrollo dispositivos móviles con WindowsCE	Con este tutorial conocerás, pantalla a pantalla, todos los pasos para crear y probar tu primera aplicación Visual C++ para Windows CE, sin ningún conocimiento.
Integración de IPlanet y Weblogic	Cesar Crespo nos muestra como instalar e integrar IPlanet y Webgic
Java en tu movil con J2ME	Os enseñamos como construir una aplicación Java capaz de correr en tu Movil gracias a J2ME

www.AdictosAlTrabajo.com Opimizado 800X600