Argentina $8,90 (recargo al interior $0,20) / México: $45

USERS

Microsoft

Curso teórico y práctico de programación

Desarrollador .net

Con toda la potencia de **Visual Basic .NET** y **C#**

La mejor forma de aprender a programar desde cero

★★★★★

Basado en el programa Desarrollador Cinco Estrellas de Microsoft

25

Trabajando con Workflows

Transacciones, CompasanteActivity / Métodos / Eventos / Consumir Web Services

SALIDA LABORAL
La mejor preparación para rendir los exámenes
GARANTIZADA

ISBN 978-987-1347-43-8
00025
9 789871 347438

RedUSERS

COMUNIDAD DE TECNOLOGIA

¡SUMATE!

EL SITIO Nº1 DE TECNOLOGIA

Noticias al instante // Entrevistas y coberturas exclusivas // Análisis y opinión de los máximos referentes // Reviews de productos // Trucos para mejorar la productividad // Registrate, participa, y comparte tus opiniones

SUSCRIBITE

SIN CARGO A CUALQUIERA DE NUESTROS **NEWSLETTERS** Y RECIBÍ EN TU CORREO ELECTRÓNICO TODA LA INFORMACIÓN DEL UNIVERSO TECNOLÓGICO ACTUALIZADA **AL INSTANTE**

INGRESÁ A
redusers.com/suscribirse-al-newsletter
¡Y REGISTRATE YA!

Foros · Encuestas · Tutoriales · Agenda de eventos · Videos · ¡Y mucho más!

redusers.com

Seguinos en:
www.facebook.com/redusers
www.twitter.com/redusers
www.youtube.com/redusersvideos

Tracking Services o Servicios de seguimiento

Uno de los principales beneficios que ofrece el motor de ejecución de Windows Workflow Foundation es que posee la capacidad de proveer servicios que tienen como objetivo simplificar las tareas comunes que debe realizar un desarrollador de flujos de trabajo.
Uno de ellos es el de **Tracking Services**, y lo podemos definir como el servicio que proporciona automáticamente el rastreo y/o seguimiento de la ejecución de cualquier instancia de un flujo de trabajo. Esta característica es particularmente importante debido a que en los flujos de trabajo muchas veces nos interesa saber qué actividades fueron ejecutadas y en general el camino que siguió el motor de ejecución para completar determinado flujo de trabajo.

Para poder acceder a los servicios que ofrece **Tracking Services** usaremos el espacio de nombres **System.Workflow.Runtime.Tracking**, que contiene las clases que nos permitirán ya sea generar nuestra propia solución de monitoreo y seguimiento de los flujos de trabajo o bien utilizar soluciones que ya vienen listas para usarse como el **SQL Tracking Service**.

Antes de continuar los adentraremos un poco en la arquitectura de **Tracking Services**.
Básicamente existen tres componentes principales dentro de la arquitectura de los **Tracking Services** y éstos son:

- Tracking Profiles (Perfiles de seguimiento).
- Tracking Runtime (Motor en tiempo de ejecución de seguimiento).
- Channels (Canales).

Tracking Profiles o Perfiles de seguimiento:

Los perfiles de seguimiento representan la manera por la que es posible identificar los orígenes y las fuentes de los eventos que deseamos capturar dentro del monitoreo del flujo de trabajo. Estos perfiles son indispensables al momento de generar nuestra propia solución de seguimiento. Existen tres tipos de eventos o sucesos a los que nos podemos suscribir para monitorearlos, y éstos son:

- Workflow Events.
- Activity Events.
- User Events.

Los **Workflow Events** o **Eventos de un flujo de trabajo** surgen a partir de una instancia y son equivalentes a los eventos a los que nos suscribimos al momento de invocar un flujo de trabajo. Como ejemplo de estos flujos tenemos los eventos: **Created**, **Terminated**, **Suspended**, etcétera.

Los **Activity Events** o **Eventos de actividades** son generados por las actividades que están siendo ejecutadas dentro de la instancia del flujo de trabajo.
Existen ocasiones en las que es necesario obtener información adicional sobre lo que está sucediendo en un determinado flujo de trabajo. Esto no necesariamente corresponde a la situación de la instancia del mismo flujo de trabajo, o de alguna de sus actividades, sino que es necesario generar un suceso personalizado para capturar determinado comportamiento.
Para cada uno de los eventos descritos, conocidos como **User Events** o **Eventos de usuario**, previamente existe una clase específica que guarda información sobre éstos. Nos referimos a las clases **WorkflowTrackingRecord ActivityTrackingRecord** y **UserTrackingRecord**, respectivamente.

Traking Runtime o Motor de tiempo de ejecución de seguimientos:

El motor de tiempo de ejecución de los servicios de seguimiento de los flujos de trabajo se encarga de iniciar los servicios de monitoreo que han sido declarados antes de la invocación de la instancia del flujo de trabajo que deseamos monitorear. Para hacer esto utiliza la información encontrada dentro de los perfiles de seguimiento.

Channels o Canales:

Los canales de monitoreo se usan para enviar los registros asociados a una instancia de un flujo de trabajo, cuando éste se encuentra en un **Tracking Point** o **Punto de seguimiento**, que es la clase genérica que recibe información de lo que sucede dentro del flujo de trabajo.

A continuación crearemos la implementación básica de un servicio de seguimiento personalizado. Para ello, crearemos un nuevo proyecto en **Visual Studio** del tipo **Sequential Workflow Console Application**, al cual le agregaremos un formulario de Windows (**Windows Forms**) que llamaremos **frmInicio**. En él colocaremos dos contendores del tipo recuadros de grupo (**GroupBox**) y en cada uno de ellos una caja de texto. A una de éstas la usaremos como receptor de la información que generan las actividades del flujo de trabajo, y a otra como receptor de los mensajes que genera el servicio de seguimiento (**Tracking Service**) que crearemos en este ejemplo (ver Figura 56). A cada control de la caja de texto lo nombraremos **txtMensaje** y **txtTracking**, respectivamente.

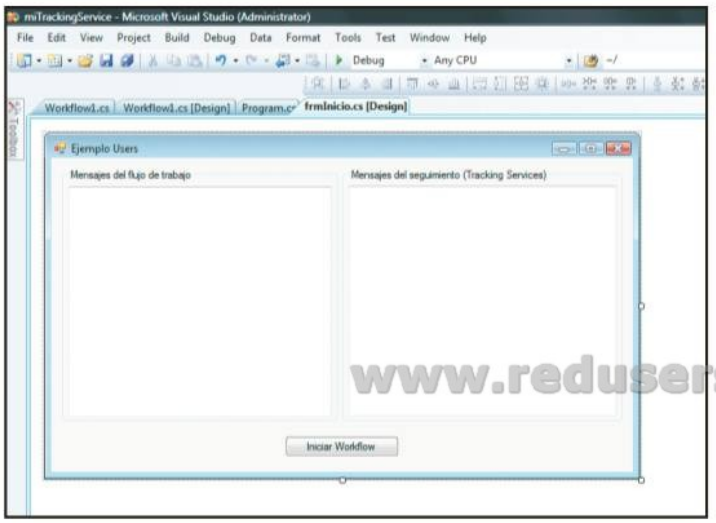


FIGURA 056 | Prototipo de nuestra aplicación con Tracking Services personalizados.

Los flujos de trabajo por lo general corren en hilos de ejecución diferentes al del host, por lo tanto, necesitamos crear métodos a los que se pueda acceder mediante delegados en nuestro formulario de Windows para así poder escribir datos en las cajas de texto.
Luego, editaremos el flujo de trabajo que incluye la solución y agregaremos una actividad del tipo **CodeActivity** (ver Figura 57). A esta actividad le especificaremos la propiedad **ExecuteCode** con el nombre "**EjecutarCodigo**", que será el método con el que escribiremos mensajes en la caja de texto.
El código en cuestión es el siguiente:

C#:

```
Program.frmInicio.EscribirMensaje("Mensaje
desde \"EjecutarCodigo\" en
\"codeActivity1\"");
```

Posteriormente, vamos a invocar la ejecución del flujo de trabajo en el manejador de evento del clic del botón del formulario de Windows y generamos la suscripción al evento **WorkflowCompleted** del flujo de trabajo con el código que aparece a continuación:

C#:

```
workflowRuntime = new WorkflowRuntime();
workflowRuntime.WorkflowCompleted += new
EventHandler<WorkflowCompletedEventArgs>
```

```
(workflowRuntime_WorkflowCompleted);
WorkflowInstance instance = workflowRuntime.
CreateWorkflow(typeof(miTrackingService.
Workflow1));
instance.Start();
```

Al momento de ejecutar el código mostrado, en la caja de texto que recibe la información generada desde flujo de trabajo deberá aparecer el mensaje "Mensaje desde "EjecutarCodigo" en "codeActivity1".
Para poder probar nuestra aplicación, debemos modificar algunas propiedades de nuestro proyecto, porque recordemos que creamos un proyecto del tipo **Console Application**. Por lo tanto, cambiaremos el tipo de aplicación desde las propiedades de nuestro proyecto, modificaremos el código de nuestro método **Main** del archivo **Program.cs**.
Por último, nuestra clase **Program** debe quedar de la siguiente manera:

C#:

```
class Program {
  public static frmInicio frmInicio;
  static void Main(string[] args) {
    frmInicio = new frmInicio();
    Application.EnableVisualStyles();
    Application.Run(frmInicio);
  }
}
```

Hemos llegado al momento en el que comenzaremos con el desarrollo del servicio de seguimiento. Añadiremos una nueva clase a nuestro proyecto, que denominaremos **ServicioSeguimiento**. Esta clase deberá heredar de la clase **TrackingService**, que es la clase abstracta base que provee la interfaz entre un servicio de seguimiento y el motor de ejecución de los servicios de seguimiento que mencionamos antes en los conceptos de arquitectura de este capítulo.

Para generar exitosamente una clase que pueda ser consumida por el motor de seguimiento es necesario sobrescribir (hacer override) los siguientes miembros de la clase base **TrackingService**:

- GetProfile(Guid).
- GetProfile(Type, Version).
- TryGetProfile.
- GetTrackingChannel.
- TryReloadProfile.

El motor de ejecución de los servicios de seguimiento solicita un objeto del tipo **TrackingChannel** para cada una de las instancias que tiene un **TrackingProfile** asignado y usa este **TrackingChannel** para mandar los registros asociados a dar una instancia de flujo de trabajo. Dentro de la definición del **TrackingProfile** se debe especificar cuáles serán los eventos que serán monitoreados y cuya información es recibida por un **TrackingRecord**, que como vimos anteriormente, puede tener como clases concretas los tipos: **ActivityTrackingRecord**, **UserTrackingRecord** o **WorkflowTrackingRecord**.

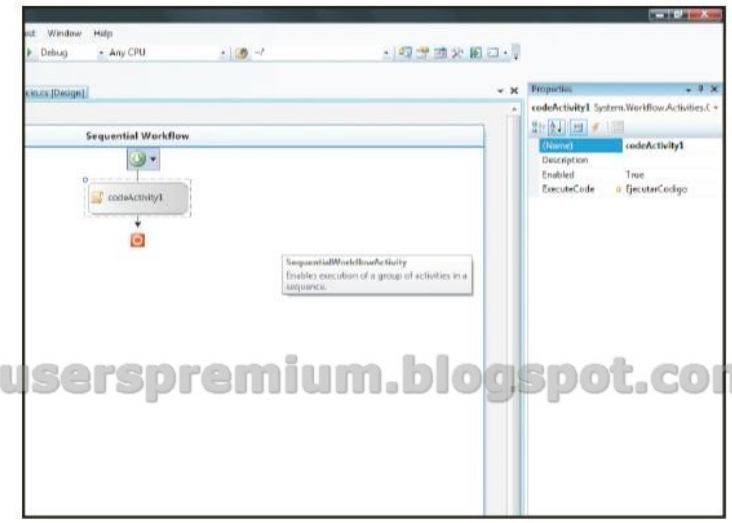


FIGURA 057 | Prototipo de nuestro flujo de trabajo que implementará los Tracking Services personalizados.

Luego, la infraestructura de WF manda a llamar al método **TryReloadProfile** para verificar si se debe recargar el **Profile**. Este proceso le permite a un cliente o a un servicio cambiar un perfil de seguimiento y hacerlo en forma dinámica.
A continuación, veremos el código de nuestro servicio se seguimiento:

C#:

```
class ServicioSeguimiento : TrackingService {
  protected override bool TryGetProfile
  (Type workflowType, out TrackingProfile
  profile) {
    profile = ObtenerProfile();
    return true;
  }

  protected override TrackingProfile
  GetProfile(Guid workflowInstanceId) {
    throw new NotImplementedException
    ("No implementado");
  }

  protected override TrackingProfile
  GetProfile(Type workflowType, Version
  profileVersionId) {
    return ObtenerProfile();
  }

  protected override bool TryReloadProfile
  (Type workflowType, Guid workflow
  InstanceId, out TrackingProfile profile) {
    profile = null;
    return false;
  }

  protected override TrackingChannel
  GetTrackingChannel(TrackingParameters
  parameters) {
    return new CanalSeguimiento
    (parameters);
  }

  private TrackingProfile ObtenerProfile() {
    TrackingProfile profile = new
    TrackingProfile();
    profile.Version = new Version("3.0.0");

    ActivityTrackPoint PuntoSeguimiento =
    new ActivityTrackPoint();
    ActivityTrackingLocation
    LocacionActividad = new
    ActivityTrackingLocation(typeof(Activity));
    LocacionActividad.MatchDerivedTypes =
    true;

    IEnumerable<ActivityExecutionStatus>
    estados = Enum.GetValues(typeof
    (ActivityExecutionStatus)) as
    IEnumerable<ActivityExecutionStatus>;
    foreach (ActivityExecutionStatus estado
    in estados) {
      LocacionActividad.
      ExecutionStatusEvents.Add(estado);
    }

    PuntoSeguimiento.MatchingLocations.
    Add(LocacionActividad);
    profile.ActivityTrackPoints.Add
    (PuntoSeguimiento);
    return profile;
  }
}
```

Como podemos ver en el código presentado, el método **GetTrackingChannel** devuelve un objeto del tipo **TrackingChannel** donde en este caso específicamente enviaremos a la forma la información que ha sido monitoreada en forma automática al momento de ser ejecutado el flujo de trabajo.

A continuación le añadiremos a nuestro programa una nueva clase denominada **CanalSeguimiento**, que heredará de **TrackingChannel** y cuyo objetivo será manejar los datos para persistirlos. En nuestro ejemplo simplemente se envía la

información recopilada a la caja de texto que se encuentra en el formulario. De esta clase es necesario sobrescribir los métodos heredados de la clase abstracta **Send** e **InstanceCompletedOrTerminated** para lograr un código como el siguiente:

C#:

```
public class CanalSeguimiento :
TrackingChannel {
  private TrackingParameters Parametros =
  null;
  protected CanalSeguimiento() {
  }
  public CanalSeguimiento
  (TrackingParameters parametros) {
    this.Parametros = parametros;
  }
  protected override void Send
  (TrackingRecord Registro) {
    ActivityTrackingRecord
    RegistroActividad =
    (ActivityTrackingRecord)Registro;

    Program.frmInicio.EscribirSeguimiento
    ("Hora: " + RegistroActividad.
    EventDateTime.ToString());
    Program.frmInicio.EscribirSeguimiento
    ("Fecha: " + RegistroActividad.
    QualifiedName.ToString());
    Program.frmInicio.EscribirSeguimiento
    ("Tipo: " + RegistroActividad.
    ActivityType);
    Program.frmInicio.EscribirSeguimiento
    ("Estado: " + RegistroActividad.
    ExecutionStatus.ToString());
  }
  protected override void InstanceCompleted
  OrTerminated() {
    Program.frmInicio.EscribirMensaje("Se
    terminó la instancia");
  }
}
```

Hemos descrito los pasos básicos para generar un servicio de seguimiento personalizado para cualquier instancia de un flujo de trabajo, que monitorea la ejecución de actividades y en este caso muestra la información a la etiqueta definida en nuestro formulario **frmInicio**.
Lo único que falta para que nuestro **Tracking Service** personalizado funcione es agregar el servicio cuando se invocan las islas de los flujos de trabajo, es decir, en el que debemos de modificar nuestra invocación con la siguiente línea de código:

C#:

```
workflowRuntime.AddService(new
ServicioSeguimiento());
```

Al momento de ejecutar el código anterior y oprimir el botón de invocación del flujo de trabajo, podemos observar cómo el motor de ejecución de Workflow Foundation ejecuta los diferentes métodos definidos en las clases **ServicioSeguimiento** y **CanaldeSeguimiento** y muestra en las cajas de texto el resultado del seguimiento del flujo de trabajo (ver Figura 58).

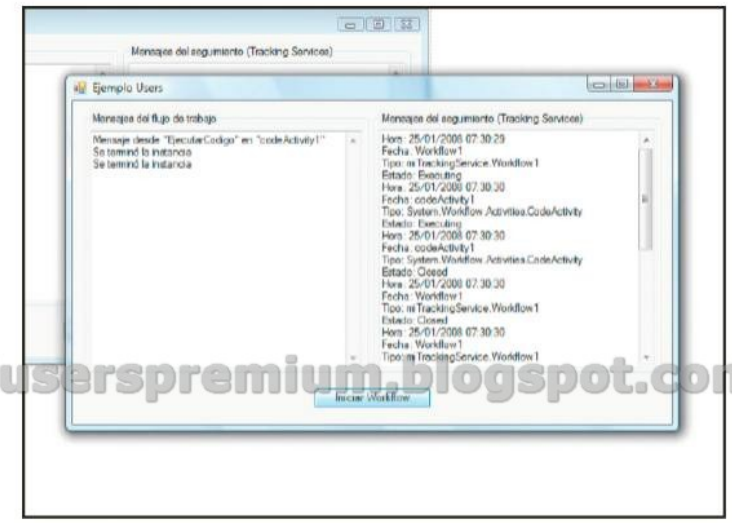


FIGURA 058 | Ejecución y resultado de nuestra aplicación con nuestro servicio de seguimiento personalizado.

Como pudimos observar en el ejemplo anterior, podemos hacer un seguimiento muy específico de todas las situaciones que tienen lugar durante la ejecución de nuestro flujo de trabajo para llevar una auditoría muy detallada de lo que sucede en el WF. Sin embargo, no siempre es necesario programar lo que hemos visto previamente, sino que WF incluye una solución ya lista para usar basada en **SQL Server** que se denomina **SQLTracking** y que hace lo mismo que ya vimos, pero su persistencia es en el servidor de base de datos SQL Server. Por ende, nosotros no tendremos que programar absolutamente nada porque todo ya está hecho. Cabe señalar que esta implementación utiliza la misma técnica y usa como clase de bases las mismas clases que utilizamos para nuestro **TrackingService** personalizado.

Para poder utilizar esta funcionalidad es necesario ejecutar unos scripts de SQL que se distribuyen con la instalación del .NET Framework 3.0, y que se encuentran en C:\WINDOWS\Microsoft.NET\Framework\v3.0\Windows Workflow Foundation\SQL\EN. En la Tabla 012 se detallan los scripts necesarios para implementar el servicio de seguimiento en SQL Server.

Después de modificar nuestro ejemplo previo, agregamos el servicio **SqlTrackingService**, que se encuentra en el espacio de nombre using **System.Workflow.Runtime.Tracking**, y que como podemos ver, contiene un constructor sobrecargado, en el que le pasaremos como parámetro la cadena de conexión que indica la instancia y el nombre de la base de datos de SQL Server.

C#:

```
workflowRuntime.AddService(new
SqlTrackingService(@"Data Source=.\sqlexpress;
Initial Catalog=WorkflowSamples;Integrated
Security=sspi"));
```

Luego de ejecutar nuestra aplicación, podemos ir al **Management Studio** de **SQL Server** y ejecutar la siguiente consulta SQL que nos mostrará la efectividad del **SqlTrackingSer-vice**. El resultado de esta consulta lo podemos observar en la Figura 59.

SQL:

```
SELECT TrackingWorkflowEvent.Description AS
Evento,
   WorkflowInstanceEvent.EventDateTime AS
   HoraInicial,
   WorkflowInstance.WorkflowInstanceId AS
   InstanciaWF,
   Type.TypeFullName AS NombreWF
FROM WorkflowInstanceEvent
INNER JOIN TrackingWorkflowEvent ON
   WorkflowInstanceEvent.
   TrackingWorkflowEventId = TrackingWorkflowEvent.
   TrackingWorkflowEventId AND
   WorkflowInstanceEvent.TrackingWorkflow
   EventId = TrackingWorkflowEvent.
   TrackingWorkflowEventId
INNER JOIN WorkflowInstance ON
```

Results | Messages

	Evento	HoraInicial	InstanciaWF	NombreWF
1	Created	2008-01-21 07:42:45.903	D378E524-29E3-4780-947A-82F955211551	WFDemos.Workflow1
2	Started	2008-01-21 07:42:45.943	D378E524-29E3-4780-947A-82F955211551	WFDemos.Workflow1
3	Completed	2008-01-21 07:42:46.263	D378E524-29E3-4780-947A-82F955211551	WFDemos.Workflow1

FIGURA 059 | Resultado de la consulta SQL sobre las tablas utilizadas por el servicio de seguimiento en SQL.

```
WorkflowInstance.WorkflowInstanceInternalId
= WorkflowInstanceEvent.
WorkflowInstanceInternalId
INNER JOIN Type ON
   Type.TypeId = WorkflowInstance.
   WorkflowTypeId
```

Persistencia

Existe otro concepto muy importante dentro de los flujos de trabajo y que también es provisto mediante los servicios de ejecución del motor de flujos de trabajo de Workflow Foundation: la persistencia. Éste se utiliza siempre que se requiere recordar el estado de una determinada instancia de un flujo de trabajo para luego ocuparla en otro momento. Los servicios de persistencia de Workflow Foundation se encargan de guardar el estado de cualquier instancia de los flujos de trabajo, y lo hacen de manera automática cuando ocurre algún evento que haga que el flujo de trabajo entre en un estado de suspensión, es decir, toda la información referente al flujo se guarda en un medio persistente.

Ahora modificaremos el proyecto previamente usado para verificar el funcionamiento de los servicios de persistencia de WF. Lo primero que haremos será modificar nuestro flujo de trabajo mediante el agregado de una actividad del tipo **SuspendActivity** y otra del tipo **CodeActivity** después de la primera, cuyo código será:

C#:

```
Program.frmInicio.EscribirMensaje("Mensaje
desde \"EjecutarCodigo2\" en
\"codeActivity2\"");
```

También modificaremos la ventana principal del proyecto (**frmInicio**) y le agregaremos una caja de texto y un botón adicional (ver Figura 60). Ahora modificaremos la invocación del flujo de trabajo en el primer botón para añadir el servicio de **SqlWorkflowPersistenceService** que se encuentra dentro del espacio de nombres **System.Workflow.Runtime.Hosting**.

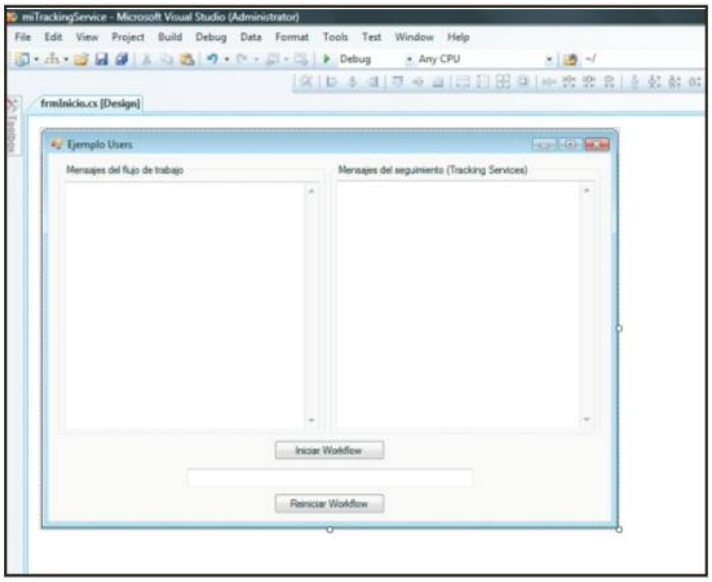


FIGURA 060 | Ejemplo de nuestra aplicación modificada para implementar los servicios de persistencia.

Tabla 12 | Implementar TrackingServices en SQL Server

Script	Descripción
SqlPersistenceService_Schema.sql	Crea la estructura de SqlPersistenceService en la Base de Datos.
SqlPersistenceService_Logic.sql	Crea la lógica de negocio de SqlPersistenceService en la Base de Datos.
Tracking_Schema.sql	Crea la estructura de SQLTracking en la Base de Datos.
Tracking_Logic.sql	Crea la lógica de negocio de SQLTracking en la Base de Datos.

Asimismo, agregaremos un manejador de eventos para que el host se suscriba al evento que generará la **SuspendActivity** y que le notificará al host de esa suspensión. El código final para la inicialización del motor de flujo de trabajo debe quedar de la siguiente forma:

C#:

```
...
  workflowRuntime.AddService(new
  SqlWorkflowPersistenceService(@"Data
  Source=.\sqlexpress;Initial Catalog=
  WorkflowSamples;Integrated Security=sspi"));
...
EventHandler<WorkflowSuspendedEventArgs>
(workflowRuntime_WorkflowSuspended);
...
```

En el método **workflowRuntime_ WorkflowSuspended** generado agregaremos el siguiente código para obtener el identificador de instancia del WF que usaremos posteriormente para solicitarle al motor de WF que reactive el flujo de trabajo persistido. Haremos esto mediante el siguiente código.

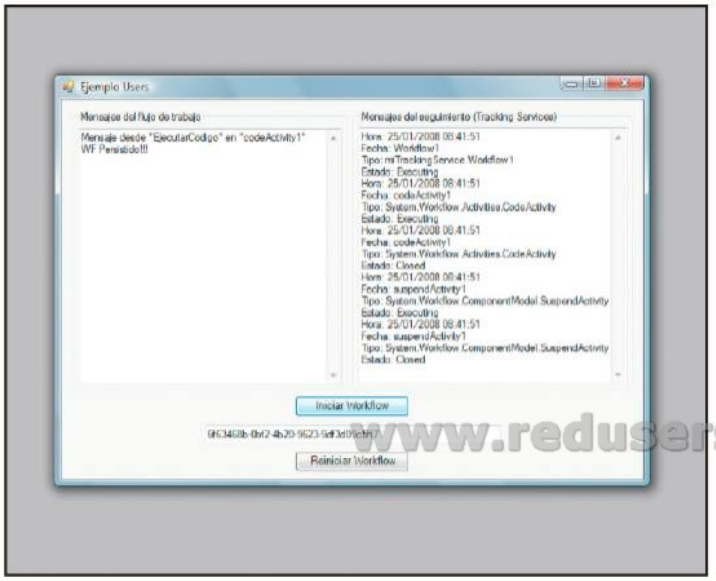


FIGURA 061 | Resultado de la ejecución de nuestro flujo de trabajo con la utilización de los servicios de persistencia.

C#:

```
e.WorkflowInstance.Unload();
this.EscribirMensaje("WF Persistido!!!");
//- Agregamos el id de instancia a caja de
texto para hacer la reactivacion
this.txtPersistence.Text = e.
WorkflowInstance.InstanceId.ToString();
```

Para finalizar, le añadiremos al nuevo botón el siguiente código que invoca a los servicios de persistencia para reactivar el flujo de trabajo previamente persistido (debido a que fue suspendido y descargado en el paso previo). Para esto utiliza como su identificador, el identificador de la instancia que se encuentra en la caja de texto.

C#:

```
workflowRuntime = new WorkflowRuntime();
workflowRuntime.AddService(new
SqlWorkflowPersistenceService(@"Data
Source=.\sqlexpress;Initial Catalog=
WorkflowSamples; Integrated Security=sspi"));
workflowRuntime.WorkflowCompleted += new
EventHandler<WorkflowCompletedEventArgs>
(workflowRuntime_WorkflowCompleted);

workflowRuntime.StartRuntime();
WorkflowInstance instance =
workflowRuntime.GetWorkflow
(new Guid(this.txtPersistence.Text));
instance.Resume();
```

Al ejecutar el flujo de trabajo podemos observar que la instancia se suspende después de que se ejecuta la primera actividad de código, y posteriormente la caja de texto adquiere el GUID que representa la instancia del WF.

Al oprimir el botón con el texto **Reactivar Workflow**, se reactivará el flujo de trabajo persistido mediante **SqlPersistenceServices** y el motor de ejecución de WF completará el proceso de ejecución del flujo de trabajo.

LA ESTRELLA 4

Cuarto repaso

Realizaremos otro repaso, esta vez de los conceptos adquiridos a lo largo de las entregas realizadas que componen la estrella 4 de nuestro curso Desarrollador .NET, con el objetivo de estar preparados para rendir el examen.

En las siguientes páginas, revisaremos lo aprendido hasta ahora. Comenzaremos con un repaso del Framework 3.0, el núcleo de las nuevas tecnologías ofrecidas por Microsoft. Continuaremos con Windows Cardspace, seguiremos con Windows Presentation Foundation, Windows Communication Foundation, y finalmente Windows Workflow Foundation.

Introducción al Framework 3.0

- En un principio, el .NET Framework 3.0 se llamó WinFX, y estaba integrado por un conjunto de cuatro APIs que son:

  - Windows Communication Foundation.
  - Windows Presentation Foundation.
  - Windows Workflow Foundation.
  - Windows CardSpace.

Windows CardSpace

Las identidades digitales son comunes al ser transmitidas en la red, y cada una está representada por algún tipo de token de seguridad. Un token o una ficha de seguridad es sólo un conjunto de bytes que expresan información sobre una identidad digital. La información que incluye consiste en una solicitud o más, cada una de las cuales lleva una parte de la identidad.

Aspectos importantes de CardSpace

- **Apoyo a cualquier sistema de identidad:** las múltiples identidades que usamos provienen de distintas fuentes y se expresan en diversas formas.
- **Conciencia del usuario y control de identidad digital:** uno de los principales objetivos de CardSpace y del metasistema de identidad es capacitar a los usuarios de todo el mundo para tomar buenas decisiones acerca del uso de sus identidades.
- **Mejora de la confianza del usuario en la autenticación:** el hecho de que los sitios web se independicen de la contraseña de inicio de sesión ayuda a reducir el problema del phishing.

Proceso de autenticación

El proceso se inicia cuando ingresamos en el área de la banca personal. En este punto, la página verifica que estemos identificados. Para hacerlo, nos redirección a una página de inicio de sesión, que

contiene un formulario de inicio de sesión estándar, es decir, los clásicos cuadros de texto que solicitan usuario y contraseña y, además, un objeto embebido mediante etiquetas XHTML que contiene la información que le indica a nuestro navegador que puede utilizar Windows CardSpace. Esta última opción hará aparecer la pantalla de WCS, y entonces podremos seleccionar la identidad que vamos a utilizar con este sitio.

Windows Presentation Foundation

WPF provee tres aspectos fundamentales al momento de construir una aplicación:

- Una plataforma unificada para interfaces de usuario: el desarrollador no necesita tener conocimientos sobre la creación de aplicaciones con GDI+ o Direct3D.
- Posibilidad de que desarrolladores y diseñadores trabajen juntos: éste es uno de los objetivos de WPF. WPF introduce un nuevo lenguaje de comunicación entre ambos perfiles, XAML. De esta forma, desarrolladores y diseñadores trabajan con diferentes herramientas, pero generan un único lenguaje común para ambos.
- Una tecnología común para aplicaciones basadas en Windows y Web: un desarrollador puede crear una aplicación para navegadores con XAML (XBAP) con WPF, y hacer que se ejecute sobre él mismo. De igual manera, puede utilizar el mismo código para crear aplicaciones WPF basadas en Windows.

XAML: concepto y definición

XAML se basa en la nueva tendencia de programación declarativa para escribir aplicaciones con WPF, según la cual se introduce un nuevo rol dentro del desarrollo de aplicaciones. Nos referimos al diseñador gráfico.

Principales espacios de nombres

System.Object: el punto de inicio para el modelo de programación en WPF es la exposición mediante el código administrado.
System.Threading.DispatcherObject: muchos de los objetos en WPF derivan del DispatcherObject, el cual provee los aspectos primordiales para la construcción de concurrencia y threading.
System.Windows.DependecyObject: en la construcción de aplicaciones WPF es preferible usar propiedades sobre métodos o eventos. Las propiedades son declarativas y nos permiten ejecutar la funcionalidad deseada de una manera mucho más sencilla. Dentro de WPF, se las conoce como un sistema de propiedades ricas, derivado del tipo DependecyObject.
System.Windows.Media.Visual: la clase Visual provee la construcción de un árbol de objetos visuales, cada uno los cuales contiene instrucciones y metadatos para ejecutar varias instrucciones (transformaciones, truncaciones, etcétera).
System.Windows.UIElement: define subsistemas de base que incluyen layouts (diseños), entradas y eventos.
System.Windows.FrameworkElement: define un conjunto de políticas y customizaciones de subsistemas en las capas iniciales.

Windows Communication Foundation

WCF es una plataforma de comunicaciones y mensajería diseñada para unificar e implementar las tecnologías de aplicaciones distribuidas existentes. Es por eso que puede interoperar con múltiples tecnologías, tales como:

- WSE 3.0.
- System.Messaging.
- NET Enterprise Services.

- Servicios Web ASP.NET (asmx).
- .NET Remoting.
- MSMQ.

Service Contract: no es otra cosa que una interfaz con la descripción del método que vamos a exponer como servicio. Podrá ser identificado por cualquier cliente y, sobre la base de esta interfaz, se podrá suscribir para poder utilizar el servicio.
Data Contract: son implementaciones de clases que pueden atravesar el canal preparado por WCF para comunicarse. Estas clases tienen la propiedad de serializarse, y de transportar información de una manera controlada y administrada por WCF.
Message Contract: son clases que definen el contenido de un mensaje SOAP. Esto es su cabecera y el cuerpo. No es muy común su uso dado que la estructura del mensaje se vuelve muy compleja.
Binding y transporte de mensajes: define cómo se comunica el end-point con el resto de las aplicaciones. Aquí se define el protocolo (TCP, SOAP, etcétera), el tipo de codificación (binaria, texto) y las directivas de seguridad (SSL o basada en mensajes).
Web Services Enhancements (WSE): para Microsoft .NET es un add-on para Microsoft Visual Studio y Microsoft .NET Framework que les permite a los desarrolladores construir servicios web basados en los estándares WS-*.

Windows Workflow Foundation

Windows Workflow Foundation puede ser usado en distintos escenarios para coordinar la interacción entre aplicaciones, personas o ambos. Soporta los modelos de workflow secuencial (sequential) y la máquina de estado (state machine).

Las plantillas Sequential Workflow Console Application y State Machine Workflow Console Application brindan el modelo de workflow para dos tipos de flujo que pueden crearse: secuencial y máquina de estado.

Definición de actividades

Una actividad es el bloque básico de construcción de un workflow en WF. Todas las actividades derivan de la clase base System.Workflow.ComponentModel.Activity.
Scheduling Service: Este servicio es el encargado de la planificación de las tareas para la ejecución de los distintos flujos de trabajo dentro del motor. A estos servicios se los conoce como DefaultWorkflowShedulerService y ManualWorkflowShedulerService.
CommitWorkBatch Services: este tipo de servicio maneja las transacciones usadas por el motor de ejecución para mantener la consistencia entre el estado interno del workflow y las fuentes de almacenamiento externo (puntos de persistencia).
Servicios de persistencia: cuando se producen ciertas condiciones de un flujo de trabajo mientras está en ejecución, el motor de Windows Workflow Foundation utiliza un servicio de persistencia con el objetivo de persistir el estado de la información del flujo de trabajo.
Servicios de seguimiento: los servicios de seguimiento de Windows Workflow Foundation fueron diseñados para permitir que los hosts observaran las instancias de flujos de trabajo durante su ejecución y capturaran los eventos que se disparan durante su ejecución.
Reglas: el motor de Reglas es un mecanismo que nos permite definir los pasos a seguir según el resultado de la evaluación de las condiciones que se especifiquen al crearlo.
Policies: se establecen con la actividad Policy, a la que se le asigna un RuleSet, que se encarga de determinar su acción.

Preguntas de ejemplo

Éstas son algunas de las preguntas que tendremos que contestar para recibir la certificación de Microsoft.

1 | ¿Qué elementos interactúan en el proceso de identificación?

A. Usuario, Identity Provider y Entidad que requiere la autenticación del usuario.
B. Usuario y Entidad que requiere la autenticación.
C. Entidad que requiere la autenticación e Identity Provider.
D. Todas las opciones anteriores son incorrectas.

Respuesta correcta A. El usuario al solicitar la autenticación mediante CardSpace. Éste último utiliza un proveedor de identidades que valida contra una entidad si el usuario es válido o no.

2 | ¿Qué tipo de información se almacena sobre los sitios que visito?

A. La información del sitio al que se le acaba de mandar la información de la tarjeta.
B. La fecha y la hora, la información del sitio y los datos que mandamos sobre la tarjeta.
C. El historial no es una de las características de CardSpace.

Respuesta correcta B. Una tarjeta de identidad puede contener varios claims de seguridad en un token. Estos claims pueden ser los datos mencionados en la respuesta o cualquier otro que nosotros guardemos en la tarjeta.

3 | ¿Con qué herramienta cuenta un diseñador de interfaces de usuario para trabajar con XAML?

A. Microsoft Office 2007.
B. Microsoft Expression Interactive Designer.
C. Microsoft Expression Web Designer.
D. Respuestas b y c son correctas.

Respuesta correcta D. Las herramientas Expressions de Microsoft son las destinadas al trabajo a nivel gráfico para WPF.

4 | ¿Qué es XAML?

A. Es el subsistema de presentación unificado para Windows.
B. XAML es un lenguaje declarativo basado en XML, optimizado para describir gráficamente interfaces de usuarios visuales ricas desde el punto de vista gráfico.
C. Es la nueva plataforma para el desarrollo de aplicaciones.
D. Ninguna opción es correcta.

Respuesta correcta B.

5 | ¿Cómo se llama la propiedad de un control que logra que éste se desvanezca?

A. Opacity.
B. Width.
C. Height.
D. Ninguna opción es correcta.

Respuesta Correcta A. Para hacer que un objeto se desvanezca, tenemos que modificar el valor de esta propiedad en los objetos de WPF.

6 | ¿Qué tipos de gráficos utiliza WPF?

A. Sólo gráficos en 2D.
B. Sólo gráficos en 3D.
C. Ninguna opción es correcta.

Respuesta correcta C. Recordemos que con WPF podemos crear gráficos vectoriales, en 2D, 3D y animaciones. Esta pregunta es capciosa.

7 | ¿Cuáles son las características de WCF?

A. Provee comunicaciones más ricas.
B. Provee fácil transporte.
C. Todas son respuestas validas.

Respuesta correcta C. Las respuestas enumeradas son algunas de las características que nos ofrece WCF.

8 | ¿Dónde se puede especificar el tipo de seguridad a utilizar?

A. Se puede especificar en un archivo de configuración o por código.
B. Se puede especificar por código.
C. Se puede especificar en el cuerpo del mensaje.

Respuesta correcta A. Si bien es posible especificar el tipo de seguridad por código, siempre es aconsejable hacerlo mediante un archivo de configuración.

Ejercicios Estrella 4

Tal como en las primeras estrellas, invitamos a practicar lo visto hasta aquí, mediante la realización de ejercicios y proponemos otros para pulir más nuestro conocimiento.

EJERCICIO 1: en la página 456 se hace referencia a cómo unir propiedades en Windows Presentation Foundation. Un ejemplo muestra cómo vincular el valor de un control Slider a la propiedad Text de una caja de texto. Modifiquemos este ejemplo para asociar el valor del control Slider a la propiedad Opacity de la ventana y así modificar la transparencia de ésta.

EJERCICIO 2: en la página 481 vimos cómo crear o generar un animación en 3D con Windows Presentation Foundation. Modifiquemos el ejemplo propuesto, agreguemos dos Sliders, en el primero asociemos la propiedad Duration y en el segundo asociemos la propiedad AccelerationRatio.

EJERCICIO 3: una vez adquiridos los conceptos sobre bindings y end-points en la página 496, sobre Windows Communication Foundations crearemos una aplicación con la tecnología que utilice un binding del tipo wsHttpBinding y netTcpBinding. Esta configuración debe estar en el archivo de configuración de la aplicación. Asimismo, crearemos otra aplicación que consumirá este servicio con uno u otro enlace.

EJERCICIO 4: de los tipos de workflows estudiados en el capítulo 15, haremos referencia al workflow del tipo Secuencial. En base al ejemplo propuesto en la página 537, extenderemos éste para trabajar con los Servicios de Seguimiento. Los datos deberán alojarse en una base de datos SQL Server. Por lo tanto, utilizaremos el servicio SqlTrackingService. Al final de nuestro flujo de trabajo agregaremos una actividad del tipo Code, que se encargará de obtener los datos de la base de datos con el script propuesto en la página 583.

EJERCICIO 5: del ejemplo propuesto en la página 539, sobre flujos de trabajo de máquina de estados, crearemos un flujo de trabajo similar que utilice los servicios de persistencia SqlPersistenceService. En cada estado del flujo de trabajo se agregará una actividad del tipo Suspend antes de la actividad del tipo SetState. El identificador de instancia deberá grabarse en una tabla en una base de datos para desde allí obtenerla y reactivar el flujo de trabajo.

Recomendación

Para profundizar más sobre los conceptos adquiridos en la estrella 4, les recomendamos visitar el sitio del Framework 3.0 de Microsoft: http://www.netfx.com/. En este sitio podrán encontrar tutoriales, documentos, foros y código de ejemplo sobre todo a lo relacionado al Framework 3.0.

El final del programa

A continuación, haremos referencia a los nuevos y actuales temas que se tratan en esta última entrega.

Hace poco tiempo, Microsoft completó el curso de Desarrollador Cinco Estrellas 2005 y liberó así la última entrega, la quinta estrella. Este nuevo capítulo en el programa DCE 2005 de Microsoft consta de 4 exámenes. Veamos de qué se tratan:

1-ASP.NET AJAX: este primer tema trata acerca de la revolución del desarrollo web, en lo que a sitios o aplicaciones web dinámicas se refiere. Introduce el concepto de AJAX, y nos lleva a conocer los nuevos controles web existentes desarrollados por Microsoft, que permiten incorporar esta funcionalidad en nuestros desarrollos web de forma sencilla. AJAX significa Asynchronous JavaScript And XML (JavaScript y XML Asincrónico), y es un conjunto de técnicas y metodologías para el desarrollo de aplicaciones web ricas en Internet (RIA, por sus siglas en inglés). Mediante la implementación de AJAX, podemos hacer que nuestras aplicaciones web actualicen su contenido de forma sectorizada, es decir, de forma parcial, que solamente una parte de la página web que el usuario esté visitando se actualice. El manejo de postbacks también podemos hacerlo con AJAX, y de esta forma crear un formulario web en el que el usuario al hacer clic en el clásico botón enviar, solamente se reemplace una parte del formulario, permitiendo así que los datos que viajen por la red sean menos y la visualización del resultado sea más rápida. Dentro de los nuevos controles podemos citar los dos más importantes:

- **ScriptManager**: que se encarga de gestionar todos los scripts de java que se ejecuten en el

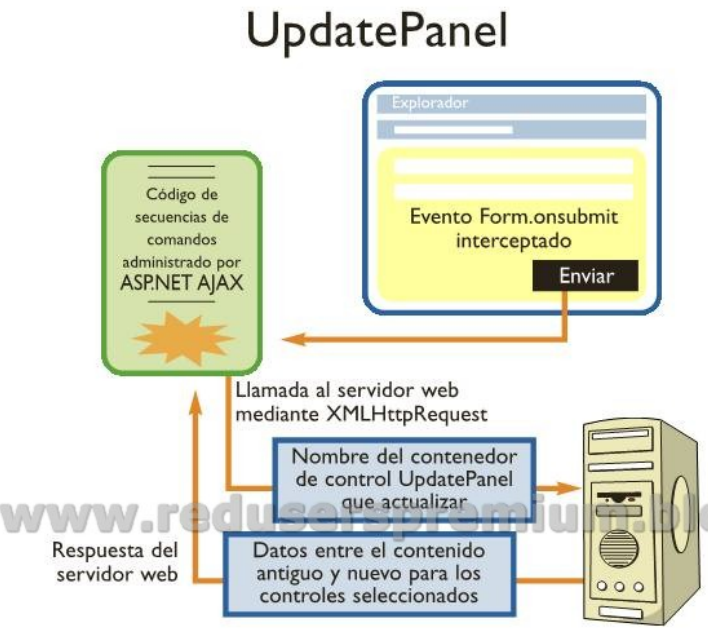


FIGURA 001 | Patrón interceptor y el control UpdatePanel.

cliente, como así también, los propios componentes AJAX de Microsoft e incluso la conexión servicios web.

- **UpdatePanel**: un panel, similar al clásico control Panel, pero con una diferencia, que todo el contenido que se muestre en él, y se actualice, se hará mediante AJAX, sin necesidad de hacer un postback completo de la página.

2-Arquitectura: en este tema, Microsoft nos introduce a los conceptos básicos sobre la Arquitectura en el mundo del desarrollo. Cómo nos ayudan las herramientas de Microsoft a definir una arquitectura y la relación estrecha que existe en cómo afrontar los desafíos de la arquitectura con Windows Workflow Foundation como herramienta. Aprenderemos los diferentes paradigmas del desarrollo de software, como la programación estructurada, no estructurada, procedural, declarativa, funcional, dirigida a eventos, entre otras. Aprenderemos el impacto de la arquitectura en la organización y sus enfoques. Conoceremos el papel del Arquitecto de la infraestructura y el modelo de optimización diseñado por Microsoft, que permite diferenciar en cuatro niveles (Basic, Standarized, Rationalized y Dynamic) el nivel de maduración de la infraestructura.

3/4-Framework 3.0: Windows Communication Foundation y Windows Workflow Foundation: finalmente, estos últimos dos temas tratados son un repaso de lo ya visto.

! Para tener en cuenta

Para más información, ingrese al portal del sitio DCE 2005 en http://www.dce2005.com y descargue los materiales que se encuentran en él. Si no está suscripto aún, es un buen momento para hacerlo y rendir los exámenes para recibir la certificación oficial de Microsoft.

X

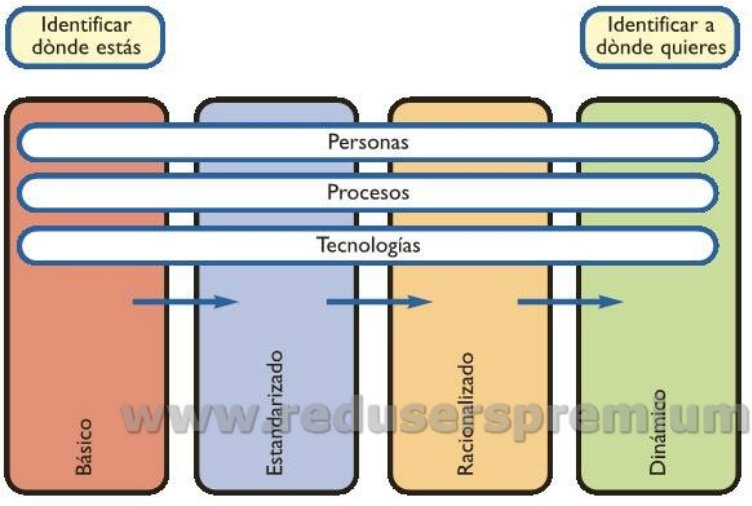
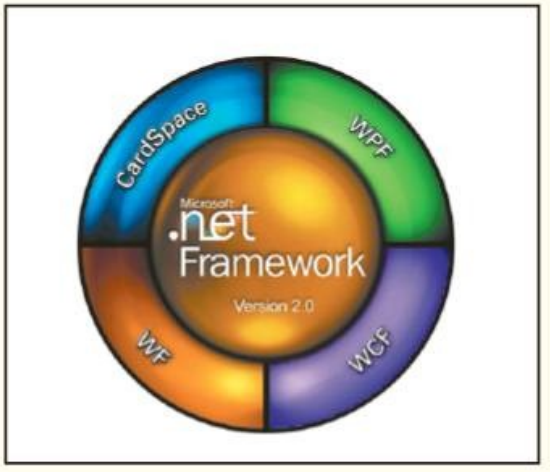
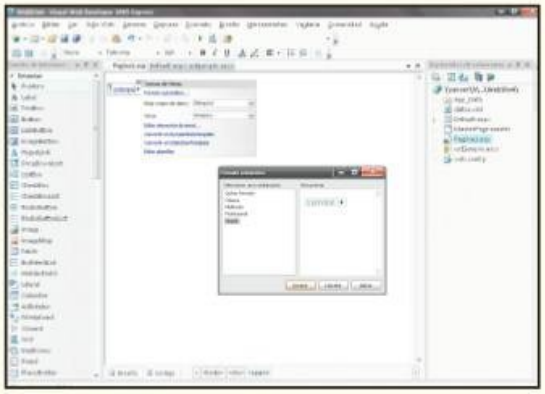


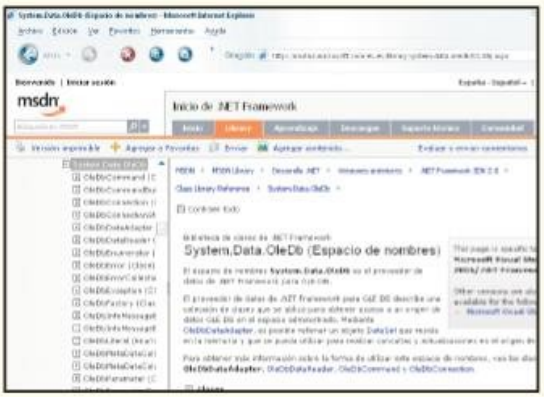
FIGURA 002 | Diagrama del Modelo de Optimización de Infraestructura (IOM) propuesto por Microsoft.

SUMARIO

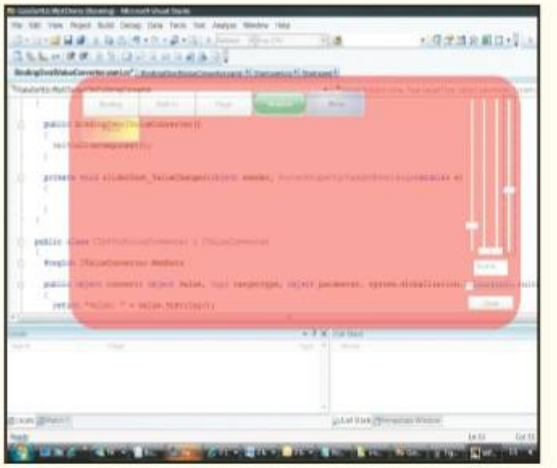


SUMARIO





SUMARIO



Argentina $8,90 (recargo al interior $0,20) / México: $45

USERS

Microsoft

Curso teórico y práctico de programación

Desarrollador .net

Con toda la potencia de **Visual Basic .NET** y **C#**

La mejor forma de aprender a programar desde cero

★★★★★

Basado en el programa Desarrollador Cinco Estrellas de Microsoft

25

Trabajando con Workflows

Transacciones, CompasanteActivity / Métodos / Eventos / Consumir Web Services

SALIDA LABORAL
La mejor preparación para rendir los exámenes
GARANTIZADA

ISBN 978-987-1347-43-8
00025
9 789871 347438