
Q-flow 3.1: Diseño de formularios personalizados

Código del manual: Qf310014ESP
Versión: 1.1
Se aplica a: Q-flow 3.1
Última revisión: 27/12/2010

Q-flow 3.1

Diseño de formularios personalizados

© Urudata Software
Canelones 1370 • Piso 2 CP11200
Montevideo, Uruguay
Teléfono: (598 2) 900 76 68 • Fax: 900 78 56

Tabla de Contenido

Introducción	4
Diseño de formularios personalizados	4
Creación de una solución vacía.....	4
Inclusión del sitio web de Q-flow en la solución.....	5
Configuración de Visual Studio para utilizar la biblioteca de controles	5
Creación de un formulario personalizado.....	7
Controles de los formularios de Q-flow	7
Recursos para hacer validaciones	8
Compatibilidad con formularios de versiones anteriores a Q-flow 3.02	12

Introducción

El propósito de este manual es explicar cómo diseñar formularios personalizados. Los formularios personalizados son páginas de ASP.NET que pueden ser utilizadas para sustituir los formularios estándar de Q-flow. El concepto de formulario personalizado se explica en el manual del diseñador de procesos del negocio. Allí también se explica cómo, una vez pronto un formulario, se debe proceder para asociarlo a un template o a un paso de un template. Este manual se ocupa solamente de explicar cómo construir un formulario personalizado.

Lectores de este manual deberían estar familiarizados con los aspectos más importantes de Q-flow y con el diseño de procesos en ese producto. Además, deberían tener conocimientos de programación en ASP.NET 2.0.

Diseño de formularios personalizados

Los formularios personalizados son páginas ASP.NET (aspx). El desarrollo de formularios personalizados requiere conocimientos de programación, ASP.NET y de la herramienta utilizada para diseñarlos (en general, Visual Studio o Visual Studio Web Developer Express Edition, que es gratuito).

La herramienta utilizada debe ser compatible con la versión 3.5 del .NET Framework, por lo que, en caso de utilizar Visual Studio, debe utilizar la versión 2008 o la versión 2010.

Para desarrollar un formulario personalizado, es conveniente que el desarrollador tenga instalado en su equipo el sitio web de Q-flow, para así poder probar fácilmente los formularios en su ambiente de desarrollo. Para obtener información acerca de cómo instalar el sitio web de Q-flow, consulte el manual de instalación y configuración (Qf310002ESP-Instalación y Configuración).

Una vez instalado el sitio web de Q-flow, cree una solución vacía en Visual Studio y agregue el sitio web de Q-flow como proyecto de la solución.

Q-flow ofrece una biblioteca de controles para utilizar en los formularios personalizados. Estos controles son los que permiten utilizar en los formularios los elementos que Q-flow incluye en los formularios estándar y tener disponibles todas las funcionalidades de los formularios estándar de Q-flow. La biblioteca de controles está implementada en el archivo Qflow.Web.CustomForms.dll.

Las siguientes instrucciones suponen que el desarrollador tiene instalado el sitio web de Q-flow en su equipo de desarrollo.

Creación de una solución vacía

1. Abra Visual Studio.
2. Haga clic en File, New Project.
3. En la ventana que le permite elegir el tipo de proyecto que desea crear, seleccione, dentro de la categoría "Other Project Types", la opción "Visual Studio Solutions".
4. Seleccione la opción "Blank Solution" (Figura 1).

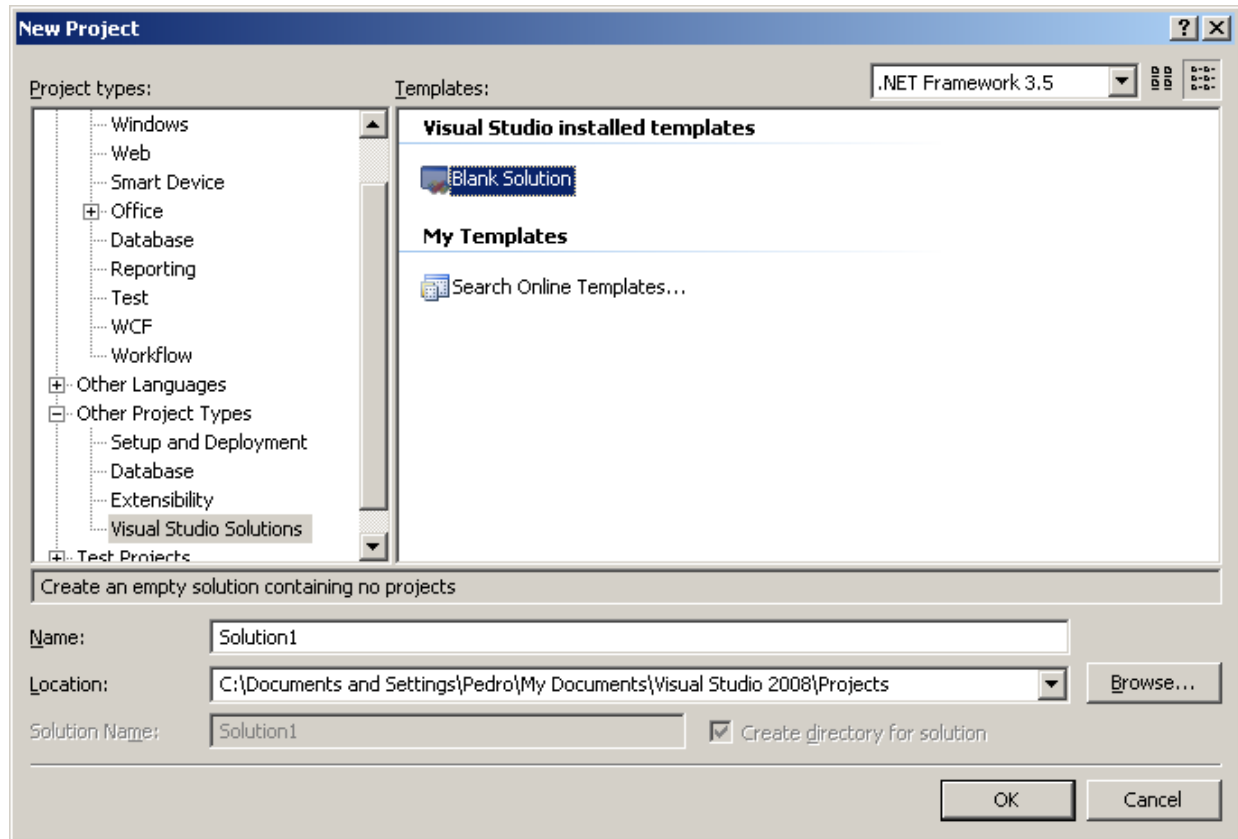


Figura 1 Creación de una solución vacía

Inclusión del sitio web de Q-flow en la solución

1. Haga clic con el botón derecho del ratón sobre el ícono que representa la solución en el explorador de soluciones.
2. Seleccione "Add", "Existing Web Site".
3. Seleccione la carpeta en la que se encuentra el sitio de Q-flow. La carpeta en la que el sitio se instala por defecto es C:\inetpub\wwwroot\QflowWebSite.

Configuración de Visual Studio para utilizar la biblioteca de controles

Antes de comenzar a diseñar los formularios, es conveniente agregar los controles de Q-flow a la barra de herramientas de Visual Studio. Para hacerlo, proceda de la siguiente forma:

1. Abra Visual Studio.
2. Seleccione en el menú "Tools" la opción "Choose Toolbox Items". Visual Studio abrirá una ventana como la que se muestra en la Figura 2.

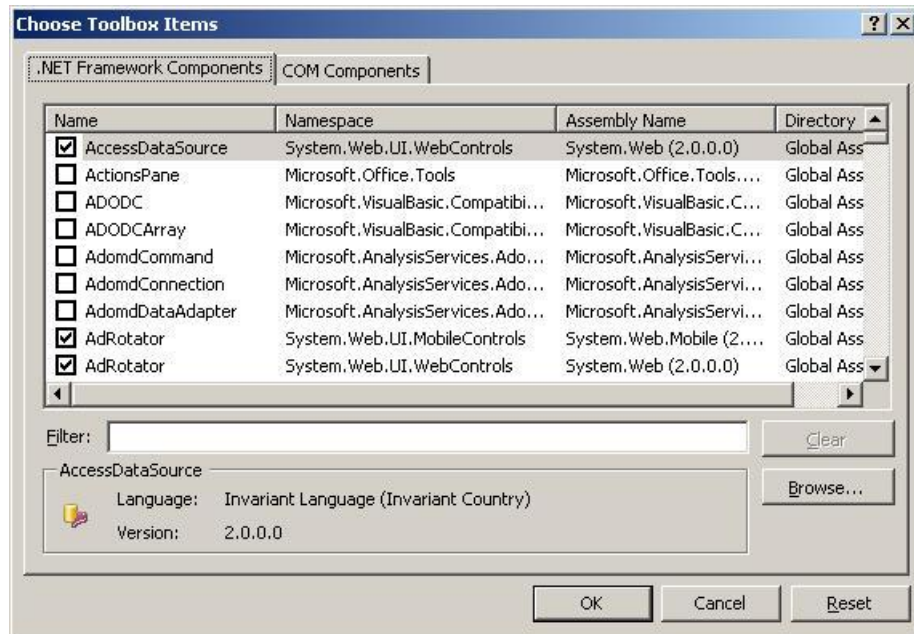


Figura 2 Ventana que permite agregar controles a la barra de herramientas de Visual Studio

3. Haga clic en "Browse..." para buscar el archivo Qflow.Web.CustomForms.dll. Navegue hasta la carpeta "bin", del sitio web, donde se encuentra el archivo, y hágale doble clic. Esto agregará los controles de Q-flow a la lista (Figura 3).
4. Haga clic en "OK". Esto agregará los controles de Q-flow. La próxima vez que abra un proyecto web en Visual Studio, los controles de Q-flow aparecerán en la barra de herramientas.

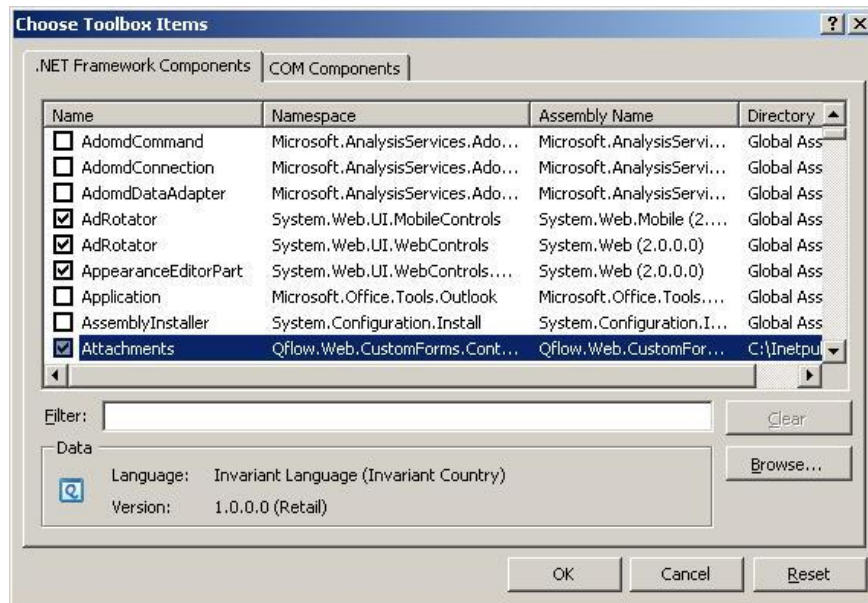


Figura 3 Los controles de los formularios de Q-flow acaban de ser agregados

Creación de un formulario personalizado

Para crear un formulario personalizado:

1. Agregue una nueva página aspx en la carpeta CustomForms (por ejemplo, MyForm.aspx).
2. Abra el archivo que contiene el código de la página y modifíquelo para que la clase de la página sea una clase derivada de alguna de las clases de los formularios de Q-flow. Cuál debe ser la clase base de su formulario depende del tipo de formulario que esté desarrollando. La tabla que se muestra abajo indica el nombre de la clase base correspondiente a cada tipo de formulario. Las tres clases base posibles pertenecen al namespace Qflow.Web.CustomForms.

Tipo de formulario	Clase base
Inicio	StartFlowBase
Respuesta	TaskResponseBase
Workflow	FlowFormBase

3. Registre en su formulario los controles de Q-flow. Si se agregaron los controles de Q-flow a la barra de herramientas, Visual Studio los registrará automáticamente la primera vez que arrastre uno de ellos hacia la vista del diseño o de edición del código ASP.NET del formulario. Los controles están en el assembly Qflow.Web.CustomForms y pertenecen al namespace Qflow.Web.CustomForms.Controls. Ejemplo:

```
<%@ Register Assembly="Qflow.Web.CustomForms"
Namespace="Qflow.Web.CustomForms.Controls" TagPrefix="qflow" %>
```

4. Desarrolle su formulario.
5. Utilice el diseñador de procesos de negocio para asociar el formulario que creó al template o paso que desee. Esta operación se describe en detalle en el manual del diseñador de procesos de negocio (Qf310003ESP-Diseñador de Procesos del Negocio).

Controles de los formularios de Q-flow

Todos los controles tienen las propiedades de la clase WebControl y otras propiedades adicionales. Además, todos los controles, salvo el control "Submit", tienen la propiedad **DisplayMode**. El valor de esta propiedad indica cómo Q-flow va a mostrar el control en el sitio web. Las opciones son cuatro:

- **ControlWithLabel:** Q-flow mostrará el control junto a una etiqueta que indicará su nombre.
- **Control:** Q-flow sólo mostrará el control, sin mostrar ninguna etiqueta.
- **Label:** Q-flow sólo mostrará la etiqueta.
- **Text:** Q-flow mostrará sólo el valor del dato representado por el control.

Los controles para formularios de Q-flow son los siguientes:

- **Data:** representa un dato de aplicación. Cuando Q-flow muestre el formulario en el sitio web, reemplazará este control por el control asociado al dominio al que pertenece el dato. El valor de la propiedad **DataName** es el nombre del dato representado por el control. Cuando Q-flow muestre el formulario, utilizará ese nombre para saber qué dato mostrar en el lugar donde está el control. Para mostrar datos que pertenecen a bloques de líneas utilice el control **Line**.

- **Role:** representa un rol del template. Cuando Q-flow muestre el formulario en el sitio web, reemplazará este control por un control que permite elegir el usuario que desempeñará ese rol, o por un control que muestra qué usuario desempeña ese rol, dependiendo de si el rol es o no editable en el paso al que está asociado el formulario. El valor de la propiedad **RoleName** es el nombre del rol. Cuando Q-flow muestre el formulario en el sitio web, utilizará ese nombre para saber qué rol mostrar en el lugar donde está el control.
- **Submit:** Es un botón. Adquiere diferente significado según el tipo de formulario donde es usado:
 - En un formulario de inicio, es el botón que permite iniciar el workflow.
 - En un formulario de respuesta, es el botón que permite responder la tarea.
 - En el formulario del workflow no tiene sentido utilizarlo.
- **TemplateVersionName:** Este control muestra el nombre de la versión del template que está siendo utilizada.
- **FlowDescription:** Este control muestra la descripción del workflow actual, o permite ingresarla, si se trata de un formulario de inicio de workflow.
- **FlowFlag:** muestra la bandera del workflow.
- **FlowImportance:** muestra la importancia del workflow.
- **FlowName:** Este control muestra el nombre del workflow actual, o permite ingresarlo, si se trata de un formulario de inicio de workflow.
- **FlowProgress:** muestra el porcentaje de avance del workflow (no confundir con el porcentaje de avance de una tarea; el porcentaje de avance del workflow se puede modificar en pasos de hito).
- **FlowStartDate:** muestra fecha y hora de inicio del workflow.
- **FlowStarterUser:** muestra el nombre del usuario que inició el workflow.
- **FlowStatus:** muestra el estado actual del workflow.
- **TaskSubject:** Este control muestra el asunto de una tarea. Sirve en los formularios de respuesta a tareas.
- **Line:** Este control define un bloque de líneas. Utilice la propiedad **LineBlock** para indicar cuál es el bloque de líneas que se desea mostrar en el formulario. Q-flow mostrará en el lugar del control todos los datos del bloque de líneas especificado.
- **TaskDescription:** Este control muestra la descripción de una tarea. Sirve en los formularios de respuesta a tareas.
- **TaskName:** Este control muestra el nombre de una tarea. Sirve en los formularios de respuesta a tareas.
- **Attachments:** Muestra los archivos adjuntos de un workflow. Si los archivos adjuntos son modificables en el paso asociado al formulario, permite agregar y borrar archivos adjuntos.
- **TemplateName:** Muestra el nombre del template al que pertenece el workflow al que está asociado el formulario.
- **TemplateVersionDescription:** Muestra la descripción de la versión del template que está siendo utilizada.
- **TemplateDescription:** Muestra la descripción del template al que pertenece el workflow al que está asociado el formulario.
- **TaskResponse:** Muestra una lista con las opciones de respuesta posibles, y permite seleccionar una respuesta.

Recursos para hacer validaciones

Q-flow provee mecanismos para interactuar con sus controles de forma que se pueda realizar validaciones del lado del cliente. Los controles de Q-flow utilizan internamente controles más sencillos (como por ejemplo, una caja de texto), y permiten acceder a esos controles internos para interactuar con ellos y realizar validaciones sobre sus contenidos. Un desarrollador puede utilizar estos

mecanismos en sus formularios personalizados. Para hacerlo, debe utilizar los scripts provistos por Q-flow.

Q-flow provee los siguientes scripts:

- **GetFlowFlagElement():** obtiene el elemento HTML que contiene la bandera del workflow.
- **GetFlowImportanceElement():** obtiene el element HTML que contiene la importancia del workflow.
- **GetFlowNameElement():** obtiene el elemento HTML que contiene el nombre del workflow.
- **GetFlowDescriptionElement():** obtiene el elemento HTML que contiene la descripción del workflow.
- **GetFlowProgressElement():** obtiene el elemento HTML que contiene el progreso del workflow.
- **GetFlowStartDateElement():** obtiene el elemento HTML que contiene la fecha de inicio del workflow.
- **GetFlowStarterUserElement():** obtiene el elemento HTML que contiene el nombre del usuario que inició el workflow.
- **GetFlowStatusElement():** obtiene el elemento HTML que contiene el estado actual del workflow.
- **GetTaskNameElement():** obtiene el elemento HTML que contiene el nombre de la tarea.
- **GetTaskDescriptionElement():** obtiene el elemento HTML que contiene la descripción de la tarea.
- **GetTaskSubjectElement():** obtiene el elemento HTML que contiene el asunto de la tarea.
- **GetTaskResponseElement():** obtiene el elemento HTML que contiene la respuesta a la tarea.
- **GetTaskProgressElement():** obtiene el elemento HTML que contiene el porcentaje de avance de la tarea.
- **GetSubmitElement():** obtiene el elemento HTML correspondiente al botón que permite contestar la tarea.
- **GetDataElement(dataName, instance):** obtiene el elemento HTML que representa el dato de aplicación cuyo nombre es igual al valor del parámetro dataName. Si el dato acepta valores múltiples, el parámetro instance indica qué línea obtener (0 corresponde a la primera línea). De lo contrario, es 0.
- **GetDataCount(dataName):** obtiene la cantidad de líneas que tiene el dato cuyo nombre coincide con el valor del parámetro dataName.
- **GetRoleElement(roleName, instance):** obtiene el elemento HTML que representa el rol cuyo nombre coincide con el valor del parámetro roleName. Si el rol acepta valores múltiples, el parámetro instance indica cuál de los valores obtener (0 corresponde al primer valor). De lo contrario, es 0.
- **GetRoleCount(roleName):** obtiene la cantidad de líneas que tiene el rol cuyo nombre coincide con el valor del parámetro roleName.
- **GetLineDataElement(lineBlock, dataName, instance):** obtiene elemento HTML que representa el dato de nombre dataName del bloque de líneas lineBlock. El parámetro instance indica qué valor se debe traer (0 si es el primer valor o si el dato no admite valores múltiples).
- **GetLineCount(lineBlock):** obtiene la cantidad de líneas que tiene el bloque de líneas indicado por el parámetro lineBlock.

En algunos casos, es posible modificar los valores de esos elementos. Por ejemplo, si el template está configurado para que un determinado dato pueda ser modificado en el paso al que corresponde el formulario, entonces es posible modificar el valor de ese dato. En otras ocasiones, no es posible modificar el valor del dato. Por ejemplo, una vez iniciado un workflow, no es posible cambiar su nombre.

Ejemplo: Asignar a un dato de tipo “Fecha” la fecha actual

Supongamos que un template tiene un dato de aplicación llamado “Fecha”, de tipo “Fecha”. Se desea que, cuando el usuario vaya a contestar ese paso, el formulario cargue automáticamente en ese dato la fecha actual. Para eso, basta con pegar el siguiente script en el formulario personalizado de ese paso:

```
<script type="text/javascript">
    attachEventCompatible(window, "load", SetCurrentDate);

    function SetCurrentDate()
    {
        var element;
        var now = new Date();
        var dateString = now.format("d/M/yyyy");

        element = GetDataElement("Fecha");
        element.value = dateString;
        element.previousSibling.value = dateString;
    }
</script>
```

La cuarta línea de código le asigna a la variable “element” el elemento que representa el dato cuyo nombre es “Fecha”. Los nombres son sensibles a mayúsculas y minúsculas. La línea siguiente le asigna al valor del dato el valor de la variable “dateString”, que contiene la fecha actual. La última línea modifica el valor que se muestra: el valor del dato es el que se guarda en element.value, pero el valor que se muestra es el que se guarda en element.previousSibling.value. La última línea no es necesaria para un dato de tipo texto. Con este script, cuando el usuario abra el formulario, en la caja de texto que muestra el valor del dato “Fecha” aparecerá la fecha actual.

Ejemplo: Cargar un texto en el nombre del workflow

NOTA: este ejemplo funciona con Internet Explorer, pero puede no funcionar correctamente en otros navegadores.

El siguiente script puede ser utilizado en un formulario personalizado de inicio para asignarle un texto (en el ejemplo “Mi flow:” seguido de la fecha actual) al nombre del workflow.

```
<script for=window event=onLoad language=vbscript>
    dim automaticFlowName
    dim flowName
    automaticFlowName = "Mi flow: " & Date
    set flowNameElement = GetFlowNameElement()
    flowNameElement.value = automaticFlowName
</script>
```

Ejemplo: Roles

NOTA: este ejemplo funciona con Internet Explorer, pero puede no funcionar correctamente en otros navegadores.

Un proceso de elaboración de informes tiene dos roles: Redactores y Revisores. Ambos admiten múltiples valores. Ningún redactor puede estar entre los revisores. El formulario de inicio de un workflow

basado en ese proceso debe impedir iniciar un workflow si algún redactor es también un revisor. El siguiente script soluciona el problema:

```
<script for=aspnetform event=onsubmit language=vbscript>
    dim cantRedactores
    dim roleElement
    'Obtiene la cantidad de miembros del rol:
    cantRedactores = GetRoleCount("Redactores")
    Dim i
    'El primer elemento corresponde a i=0:
    For i=0 To cantRedactores - 1
        'Obtiene el elemento que está en la posición i:
        set roleElement = GetRoleElement("Redactores",i)
        If EsRevisor(roleElement) Then
            'Muestra mensaje de error,
            'roleElement.nextSibling.value contiene el nombre del
miembro del rol
            MsgBox("ERROR: El redactor " & roleElement.nextSibling.value
& " es también revisor.")
            'Cancela el evento y no envía el formulario
            window.event.returnValue = false
            Exit For
        End If
    Next
</script>
<script language=vbScript>
function EsRevisor(roleElement)
    dim cantRevisores
    cantRevisores = GetRoleCount("Revisores")
    Dim i
    For i=0 To cantRevisores - 1
        If roleElement.value = GetRoleElement("Revisores",i).value Then
            EsRevisor = true
            exit function
        End If
    Next
    EsRevisor = false
end function
</script>
```

El script debe ejecutarse cuando el usuario hace clic en el botón “Iniciar”, que es el que hace que el formulario sea enviado al servidor. El atributo “for=aspnetform” especifica que el script será aplicado al formulario, y el atributo “event=onsubmit” especifica que será ejecutado cuando el usuario dé la orden de enviar el formulario, o sea, cuando haga clic en el botón “Iniciar”.

El script recorre los miembros del rol “Redactores” y verifica para cada uno de ellos si es también un miembro del rol “Revisores”. Si encuentra alguno en esas condiciones, muestra un mensaje de error y cancela el evento, evitando que se envíe el formulario. El mensaje de error muestra el nombre del usuario que es miembro de ambos roles.

Nótese que para obtener el nombre de un miembro de un rol se utiliza la expresión “roleElement.nextSibling.value”. Sin embargo, la función “EsRevisor” utiliza la expresión “roleElement.value” para comparar dos usuarios. Esto es porque roleElement.value contiene el

identificador del usuario, mientras que `roleElement.nextSibling.value` contiene el nombre del usuario, que es lo que se muestra en el formulario.

Compatibilidad con formularios de versiones anteriores a Q-flow 3.02

Los formularios personalizados realizados para funcionar con versiones anteriores a Q-flow 3.02 (Qflow 3.0 y Q-flow 3.01) no son compatibles con Q-flow 3.02 y versiones posteriores, pero basta con hacerles unas pocas modificaciones para que funcionen con la nueva versión:

- Si un formulario incluye el namespace `Qflow.Common.Exceptions` en el *code behind*, sustitúyala la sentencia de importación por una que haga referencia a `Qframework.Common.Exceptions`.
- Los controles estándar de Q-flow, que estaban en el componente `Qflow.Web.dll`, en el namespace `Qflow.Web.Controls`, ahora están en el componente `Qframework.Web.dll`, en el namespace `Qframework.Web.Controls`. Si algún formulario los utiliza en el *markup*, modifique las referencias.
- En el *code behind*, todas las referencias que se hagan al objeto `Interaction` deben ser cambiadas por referencias a `FlowInteraction`. En los lugares en los que esa nueva propiedad es utilizada, hay que importar el namespace `Qframework.Web.Interaction`.
- La mayoría de las enumeraciones (*enums*) fueron movidas del namespace `Qflow.Common` a `Qframework.Common`, por lo que, en caso de utilizar alguna, debe cambiar la referencia.
- En general, en los raros casos en los que los formularios utilicen alguna otra funcionalidad de Q-flow, si el formulario no encuentra una clase de Q-flow, busque una clase con el mismo nombre en un namespace similar cuyo nombre empiece con "Qframework".

Adicionalmente, debido a una mejora del comportamiento del control "Submit" de Q-flow, surgen algunos cambios en lo que respecta a validaciones del lado del cliente en formularios personalizados.

En la versión anterior era posible agregar rutinas de validación javascript a eventos como el "onsubmit" del formulario o el "onclick" del botón "Submit" de Q-flow, mediante *snippets* como los siguientes:

- `document.forms[0].attachEvent("onsubmit",validarForm)`
- `GetSubmitElement().attachEvent("onclick",validarForm)`

A partir de Q-flow 3.02, estas validaciones, si bien se ejecutarán, no detendrán el *postback* de la página. Sin embargo, es posible lograr que estas validaciones escritas en javascript sean ejecutadas normalmente dentro del ciclo de la página. La clave es utilizar los controles de validación de ASP.NET, especificando que se utilizará una rutina de validación del lado del cliente. Para ello, haga lo siguiente:

1. En el *markup* de la página, agregue el control de validación ASP.NET con una definición como la siguiente:

```
<asp:CustomValidator ID="CustomValidator1" runat="server" ClientValidationFunction="validarForm"
EnableClientScript="true"
ValidationGroup="QCommandButtonValidationGroup"></asp:CustomValidator>
```

2. Modifique la rutina de validación javascript para que pueda recibir los argumentos requeridos por el framework de validación de ASP.NET. En este ejemplo, la firma sería la siguiente:

function validarForm(source, clientside_arguments)

3. Dentro de la rutina de validación javascript se utiliza la propiedad booleana `clientside_arguments.IsValid` para especificar si la validación fue exitosa o no. Si la validación no es exitosa no se realiza el *postback*.