
Q-flow 3.0: Diseño de formularios personalizados

Código del manual: Qf304013ESP
Versión: 1.0
Se aplica a: Q-flow 3.04
Última revisión: 11/5/2009

Q-flow 3.0

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
Configuración de Visual Studio 2005 para utilizar la biblioteca de controles	5
Controles de los formularios de Q-flow	7
Recursos para hacer validaciones	8
Compatibilidad con formularios de versiones anteriores	11

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 aspx. Tienen código HTML y están asociados a un archivo con código C#. El desarrollo de formularios personalizados requiere conocimientos de programación, HTML y de la herramienta utilizada para diseñarlos (en general, Visual Studio 2005 o Visual Web Developer 2005 Express Edition, que es gratuito).

La clase correspondiente a un formulario de inicio de workflow debe heredar de la clase `Qflow.Web.CustomForms.StartFlowBase`. La clase correspondiente a un formulario de respuesta debe heredar de la clase `Qflow.Web.CustomForms.TaskResponseBase`. La clase correspondiente a un formulario de un workflow debe heredar de la clase `Qflow.Web.CustomForms.FlowFormBase`.

Q-flow viene con un formulario de inicio de ejemplo (`StartFlowSample.aspx`), que está dentro de la subcarpeta `CustomForms` de la carpeta del sitio de Q-flow. Dentro de esa carpeta también hay otros formularios de ejemplo.

Para construir un formulario personalizado en base a un formulario ya existente, haga lo siguiente:

1. Copie los dos archivos del formulario existente (el archivo con extensión `aspx` y el archivo con extensión `cs`) y renómbrelos.
2. Abra ambos archivos en Visual Studio 2005 u otra herramienta adecuada.
3. Modifique el atributo "CodeFile" de la página `aspx` para que contenga el nombre del nuevo archivo con extensión `cs`.
4. Aplique sobre ambos archivos las modificaciones que juzgue necesarias.
5. Cópielos en la subcarpeta `CustomForms` de la carpeta del sitio de Q-flow. Recuerde que, en el template, deberá crear un formulario personalizado cuya propiedad URL tenga una referencia a ese formulario, y asignarlo al paso que desee.

Q-flow ofrece una biblioteca de controles para diseñar 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`.

Configuración de Visual Studio 2005 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 2005. Para hacerlo, proceda de la siguiente forma:

1. Si el equipo donde trabajará no tiene instalado el sitio web de Q-flow, copie el archivo Qflow.Web.CustomForms.dll del servidor del sitio web. Este archivo se encuentra en la subcarpeta "bin" de la carpeta donde está instalado el sitio web de Q-flow. Por defecto, la carpeta donde se encuentra el archivo es la carpeta C:\Inetpub\wwwroot\QflowWebSite\bin del servidor del sitio web.
2. Abra Visual Studio 2005.
3. Seleccione en el menú "View" la opción "Toolbox". Esto hará que Visual Studio abra la barra de herramientas.
4. Haga clic con el botón derecho sobre la barra de herramientas. Visual Studio mostrará un menú contextual. Seleccione la opción "Choose ítems...". Visual Studio abrirá una ventana como la que se muestra en la Figura 1.

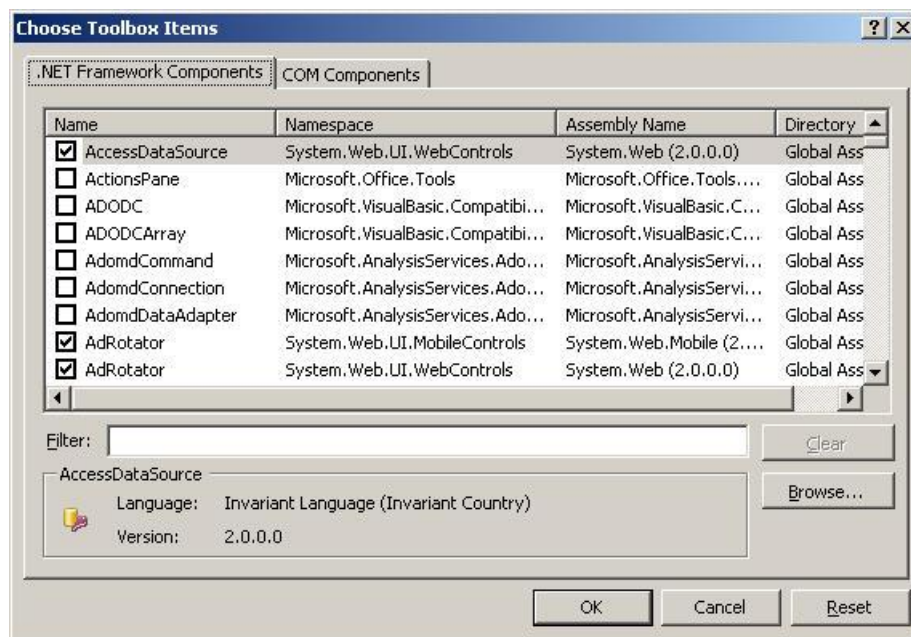


Figura 1 Ventana que permite agregar controles a la barra de herramientas de Visual Studio

5. Haga clic en "Browse..." para buscar el archivo Qflow.Web.CustomForms.dll. Navegue hasta la carpeta donde se encuentra el archivo y hágale doble clic. Esto agregará los controles de Q-flow a la lista (Figura 2).
6. 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).

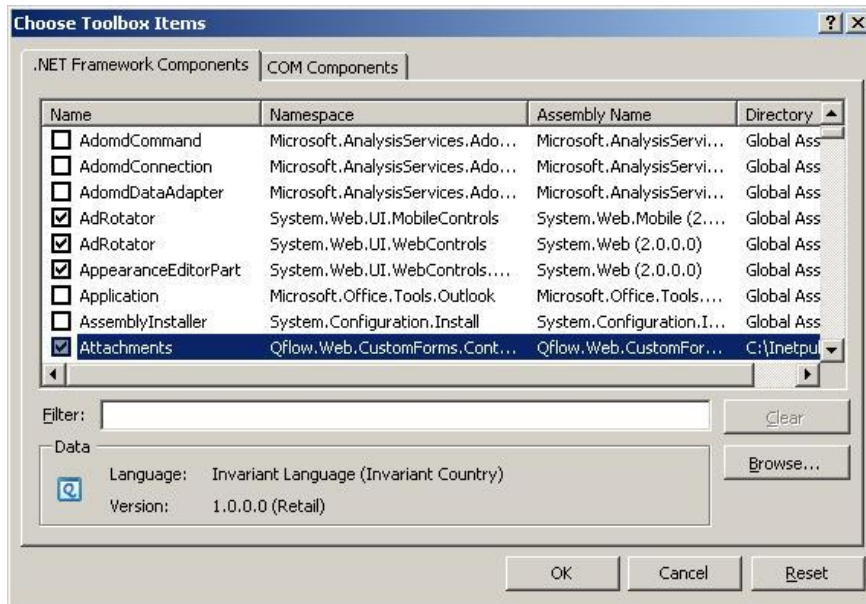


Figura 2 Los controles de los formularios de Q-flow acaban de ser agregados

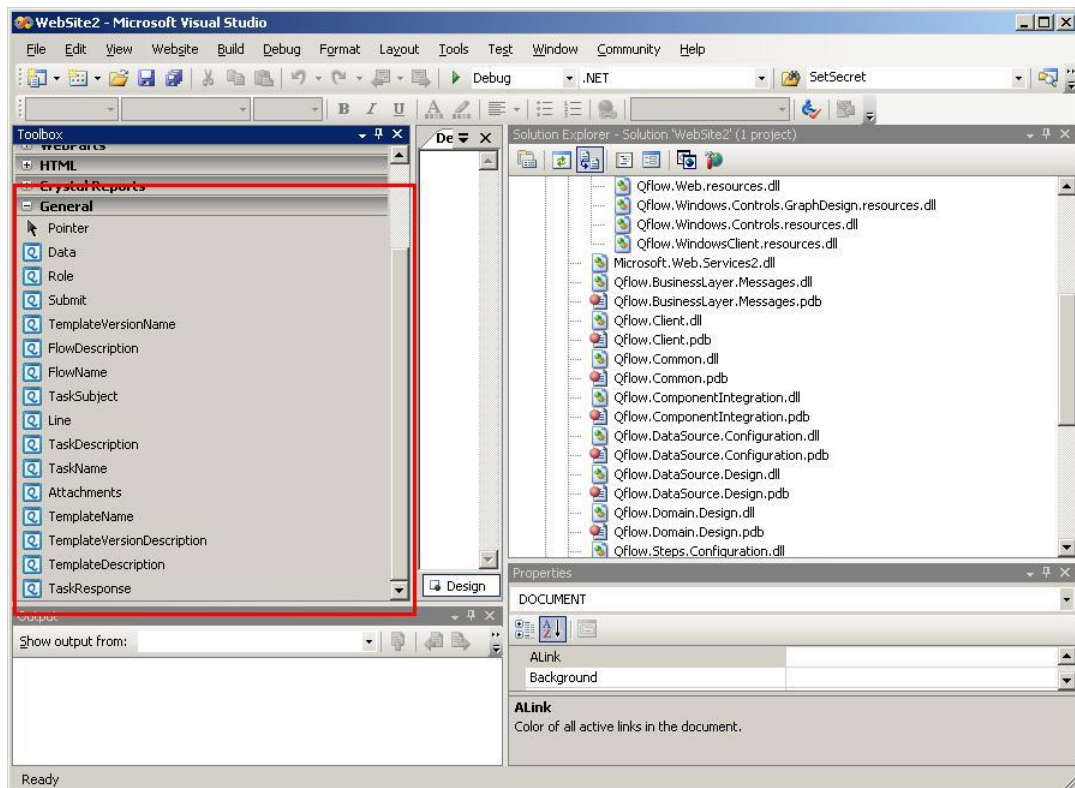


Figura 3 Barra de herramientas de Visual Studio con los controles de Q-flow

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.
- **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.
- **FlowName:** Este control muestra el nombre del workflow actual, o permite ingresarlo, si se trata de un formulario de inicio de 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. Sirve para agrupar controles Data que hacen referencia a datos de bloques de líneas.
- **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 en el archivo ElementValidationHandlerScript.js, que se encuentra en la carpeta del sitio web de Q-flow.

Ese archivo provee las siguientes funciones:

- **GetFlowNameElement():** obtiene el elemento HTML que contiene el nombre del workflow.
- **GetFlowDescriptionElement():** obtiene el elemento HTML que contiene la descripción 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 de proceso 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 for=window event=onLoad language=vbscript>
  dim dateString
  dim element
  dim nombreDatoTipoFecha
  nombreDatoTipoFecha = "FECHA"
  dateString = FormatDateTime(Date(),2)
  set element = GetDataElement(nombreDatoTipoFecha)
  element.value = dateString
  element.nextSibling.childNodes(0).value = dateString
</script>
```

La primera línea después de las declaraciones le asigna a la variable "nombreDatoTipoFecha" el nombre del dato (o sea, "FECHA"). Los nombres son sensibles a mayúsculas y minúsculas. En la siguiente línea se obtiene la fecha actual, que es almacenada en la variable dateString. Luego se utiliza la función GetDataElement para asignarle a una variable el elemento que representa el dato "FECHA" en la pantalla. Finalmente, se le asigna al valor del elemento que representa el dato el valor de la variable "dateString". 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.nextSibling.childNodes(0). 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

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

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

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*.