

Diseño de formularios personalizados

06/01/2020

Q-flow 4.3

Tabla de Contenido

Introducción	4
Vista personalizada	4
Diseño del formulario personalizado.....	5
Acceso a elementos del proceso del lado del cliente	6
Publicación del formulario personalizado	6
Área personalizada	6
Referencias	7
Registrar el área.....	7
Controladores	9
Formulario de inicio de proceso.....	9
Formulario del proceso	11
Formulario de edición del proceso.....	11
Formulario de respuesta (tarea, pregunta)	13
Modificación de _ViewStart.cshtml.....	14
Compilar los archivos y publicarlos en el sitio web	15
Asociar los formularios a la plantilla	15
Agregar los formularios personalizados	16
Configuración de los formularios personalizados.....	18
Asociar los formularios a los elementos correspondientes.....	20
Formulario personalizado de inicio y de respuesta	20
Formulario del proceso y Formulario de edición del proceso	22
Referencia de <i>helpers</i>.....	22
Referencia de la biblioteca JavaScript.....	24

Host	25
Data	26
Line	27
Role	28

INTRODUCCIÓN

El propósito de este manual es explicar cómo diseñar formularios personalizados para sustituir los formularios estándar que provee Q-flow con el sitio web. El concepto de formulario personalizado se explica en el manual del Diseñador Web de Procesos del Negocio. Allí también se explica cómo, una vez pronto un formulario, se debe proceder para asociarlo a una plantilla o a un paso de una plantilla. 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 MVC, que es la tecnología con la cual está construido el sitio web de Q-flow.

Hay dos formas de construir formularios personalizados para Q-flow:

- **Vista personalizada:** es la forma más sencilla, y consiste en diseñar una página HTML para que sustituya el formulario estándar, de modo que éste tenga una apariencia distinta.
- **Área personalizada:** se definen un área que contendrá el controlador y las vistas de un proceso o de un conjunto de procesos relacionados. El área contiene vistas para todos los formularios de estos procesos. Como en este caso se programa el controlador, esta opción implica más trabajo, pero también más personalización y flexibilidad.

Este manual viene acompañado de un ejemplo que se puede abrir utilizando Visual Studio. Este ejemplo se puede descargar en la siguiente dirección: "<https://www.urudatasoftware.com/qflow/listadoManuales.html>".

VISTA PERSONALIZADA

Construir una vista personalizada consiste en definir una vista que utiliza un modelo que es provisto por componentes de Q-flow. Qué modelo se utiliza depende del tipo de formulario que se está construyendo. Para formularios de proceso, de edición de proceso y de inicio de proceso se utiliza el modelo *Qflow.Web.Mvc.Models.FlowModel*, que representa un proceso. Para formularios de respuesta (tareas, preguntas) se utiliza el modelo *Qflow.Web.Mvc.Models.TaskModel*, que representa una tarea.

Estos modelos contienen los datos de aplicación, roles, adjuntos y datos del proceso (nombre, descripción, plantilla, etc).

Tipo de formulario	Modelo
Formulario de Inicio	<i>Qflow.Web.Mvc.Models.FlowModel</i>
Formulario de Respuesta (tarea, pregunta)	<i>Qflow.Web.Mvc.Models.FlowModel</i>
Formulario del Proceso	<i>Qflow.Web.Mvc.Models.FlowModel</i>
Formulario de Edición del Proceso	<i>Qflow.Web.Mvc.Models.TaskModel</i>

Q-flow también provee dos *layouts* para utilizar en las vistas personalizadas. Estos *layouts* incluyen el código JavaScript necesario para que los formularios funcionen correctamente. Se recomienda no modificarlos. Los *layouts* disponibles, que se pueden encontrar en el sitio de Q-flow, son:

- **~/Views/Shared/_CustomFormsLayout.cshtml:** incluye el menú lateral y el cabezal del sitio de Q-flow.
- **~/Views/Shared/_CustomFormsEmptyLayout.cshtml:** no incluye el menu lateral ni el cabezal del sitio de Q-flow.

Diseño del formulario personalizado

El formulario personalizado es un archivo cshtml, como es usual para vistas de ASP .NET MVC. Para incluir en el formulario datos de aplicación, roles, y datos del sistema como el nombre del proceso, se utiliza un conjunto de *helpers* disponibles a través de Q-flow. Estos *helpers* extienden la funcionalidad de la biblioteca de ASP .NET MVC. El formulario debe contener un elemento de tipo *form* con el id "submitForm". Dentro de este elemento se debe definir el formulario, utilizando los *helpers*. Una referencia de estos *helpers* se puede encontrar en este mismo manual, en "Referencia de *helpers*". Fuera de ese elemento se deben colocar los *helpers* que permiten incluir en el formulario los archivos adjuntos del proceso.

Acceso a elementos del proceso del lado del cliente

Q-flow también provee un conjunto de funciones en JavaScript para acceder a los datos del proceso del lado del cliente. Estas funciones están disponibles a través del objeto *Host*, que es accesible desde código JavaScript. Consulte la “Referencia de la biblioteca JavaScript” por detalles sobre las funciones disponibles.

Publicación del formulario personalizado

Una vez listo el formulario, se debe copiar en el servidor web que alberga el sitio de Q-flow. Dentro de la carpeta que contiene el sitio, hay una carpeta *Areas/CustomForm/Views/Flow*.

Por ejemplo, si el sitio está en *C:\inetpub\wwwroot\Qflow*, el archivo del formulario se debe copiar en *C:\inetpub\wwwroot\Qflow\Areas\CustomForms\Views\Flow*.

Dentro de esa carpeta se debe copiar el formulario. No olvide, si el formulario utiliza otros archivos, copiarlos también donde corresponda.

También debe especificar el formulario en el Diseñador Web de Procesos del Negocio, y asociarlo al elemento al que corresponda (al paso al que corresponde, si es un formulario de respuesta, por ejemplo). Esto se describe en “Asociar los formularios a la plantilla”.

ÁREA PERSONALIZADA

En el caso de un área personalizada, además de las vistas hay que programar el controlador. Para ello se recomienda crear un proyecto ASP .NET MVC en Visual Studio.

Los pasos para crear un área son los siguientes (detalles más abajo):

1. Crear un proyecto ASP .NET MVC en Visual Studio. El nombre del proyecto debe contener el texto “Custom”, por ejemplo **SampleCustomForms**.
2. Agregar referencias a los componentes de Q-flow necesarios para construir los formularios (ver “Referencias”).
3. Registrar el área (ver “Registrar el área”).
4. Programar uno o más controladores, con las acciones necesarias (ver “Controladores”).
5. Crear una vista por formulario personalizado. Las vistas se construyen de la misma forma que una vista personalizada común (ver “Diseño del formulario personalizado”).

6. Modificar `_ViewStart.cshtml` para contemplar los dos tipos de *layout* (ver “Modificación de `_ViewStart.cshtml`”).
7. Compilar el proyecto y publicar los archivos en el sitio web de Q-flow (ver “Compilar los archivos y publicarlos en el sitio web”).
8. Asociar el área a la versión de la plantilla para la cual fue construida.

Referencias

Una vez creado el proyecto, agregue las siguientes referencias:

- Qflow.Web.Mvc.Site.dll
- Qflow.Web.Mvc.dll
- Qframework.Web.Mvc.dll

Estos archivos se pueden encontrar en el sitio web de Q-flow. En general, es recomendable tener instalado el sitio web de Q-flow en el equipo en el que se desarrollan los formularios, para facilitar las pruebas.

Registrar el área

Para registrar el área se debe construir una clase. Por ejemplo, si está haciendo los formularios para la plantilla “Solicitud de cliente”, puede crear la clase “SolicitudClienteAreaRegistration”. Ésta debe ser una clase derivada de la clase `CustomFormsAreaRegistration`. Cuando especifique los formularios personalizados en el Diseñador Web de Procesos del Negocio, deberá especificar el nombre del área, que es el que se define en esta clase, en la propiedad `AreaName` (ver Figura 1).

```
using System.Web.Mvc;
using Qflow.Web.Mvc.Site.Areas.CustomForms;

namespace SampleCustomForm
{
    public class SampleCustomFormAreaRegistration : CustomFormsAreaRegistration
    {
        public override string AreaName
        {
            get { return "SampleCustomForm"; }
        }

        public override void RegisterArea(AreaRegistrationContext context)
        {
            RegisterRoutes(context);
        }

        private void RegisterRoutes(AreaRegistrationContext context)
        {
            context.MapRoute(
                "SampleCustomForm",
                "SampleCustomForm/{controller}/{action}/{id}",
                new { action = "Index", id = UrlParameter.Optional }
            );
        }
    }
}
```

Figura 1 Clase para registrar el área

Controladores

Las acciones que deben estar presentes en los controladores dependen de qué formularios se deseen personalizar. Hay cuatro tipos de formularios posibles:

- Formulario de inicio de proceso
- Formulario del proceso
- Formulario de edición del proceso
- Formulario de respuesta (tarea, pregunta)

En general, cada formulario que se desea personalizar requiere que se implemente al menos una acción: la que devuelve la vista que representa el formulario. Los formularios desde los cuales se puede modificar datos en Q-flow requieren, además, una acción adicional, que es la que se ejecuta cuando el usuario hace clic en el botón que dispara la modificación de los datos (por ejemplo, cuando hace clic en el botón que iniciar un proceso o cuando hace clic en el botón que responde una tarea). El formulario de inicio tiene, además, una acción adicional para la vista que se muestra al hacer clic en el paso de inicio de un proceso que ya fue iniciado. Todas las acciones correspondientes a un mismo formulario tienen el mismo nombre: sus firmas sólo deben diferir en sus parámetros.

A continuación se explica, para cada tipo de formulario, qué acciones se deben agregar a los controladores, y se da un ejemplo de cada acción.

Formulario de inicio de proceso

Un formulario de inicio personalizado requiere tres acciones. Note que todas tienen el mismo nombre, pero distintos parámetros.

- **Mostrar el formulario de inicio:** se ejecuta cuando un usuario hace clic en una plantilla para iniciar un proceso. La Figura 2 tiene un ejemplo.
- **Iniciar proceso:** se ejecuta cuando un usuario hace clic en el botón que inicia el proceso. La Figura 3 tiene un ejemplo.
- **Mostrar el formulario del paso de inicio:** se ejecuta cuando, con el proceso ya iniciado, un usuario hace clic en el paso de inicio para ver los datos de éste. La Figura 4 tiene un ejemplo.

```
[HttpGet]
[RequireRequestValue("templateId")]
public ActionResult StartTutorial(Guid templateId, Guid? versionId)
{
    FlowModel model = GetNewFlow(templateId, versionId);
    return View(model);
}
```

Figura 2 Acción para mostrar el formulario de inicio.

```
[HttpPost]
[ValidateInput(false)] //Needed for rich text box
public ActionResult StartTutorial(Guid templateId,
[ModelBinder(typeof(FlowModelBinder))] FlowModel flow)
{
    try
    {
        Guid flowId = StartFlow(flow);
        TempData["TypeMsg"] = "success";
        TempData["Msg"] = "Flow iniciado correctamente";
        return Redirect(Url.Action("Display", new { area = "", flowId =
flowId }));
    }
    catch (Exception ex)
    {
        TempData["TypeMsg"] = "error";
        TempData["Msg"] = ex.Message; //TODO: correct message and view
        return View(flow);
    }
}
```

Figura 3 Acción para iniciar un proceso

```
[HttpGet]
[RequireRequestValue("flowId")]
public ActionResult StartTutorial(Guid flowId)
{
    FlowModel model = GetStartedFlow(flowId);
    return View(model);
}
```

Figura 4 Acción para mostrar los datos de un proceso iniciado

Formulario del proceso

Un formulario del proceso requiere solamente una acción, que devuelve la vista que contiene el formulario con un modelo que contiene los datos del proceso. La Figura 5 tiene un ejemplo.

```
[HttpGet]
public ActionResult DisplayTutorialInfo(Guid flowId)
{
    FlowModel model = GetFlow(flowId); /*Get only details*/
    return View(model);
}
```

Figura 5 Acción para mostrar el formulario del proceso

Formulario de edición del proceso

Un formulario de edición del proceso requiere dos acciones:

- **Mostrar el formulario del proceso:** se ejecuta cuando un usuario hace clic en el sitio para ver el formulario del proceso. Devuelve la vista correspondiente a ese formulario. La Figura 6 tiene un ejemplo.
- **Guardar cambios:** se ejecuta cuando el usuario hace clic en el botón que permite guardar los cambios hechos en el formulario de edición del proceso. La Figura 7 tiene un ejemplo.

```
[HttpGet]
public ActionResult EditTutorialInfo(Guid flowId)
{
    FlowModel model = GetEditFlow(flowId);
    return View(model);
}
```

Figura 6 Acción para mostrar el formulario del proceso

```
[HttpPost]
[ValidateInput(false)] //Needed for rich text box
public ActionResult EditTutorialInfo([ModelBinder(typeof(FlowModelBinder))]
FlowModel flow)
{
    try
    {
        EditFlow(flow);
        TempData["TypeMsg"] = "success";
        TempData["Msg"] = "Flow editado correctamente";
        return Redirect(Url.Action("Display", new { area = "", flowId =
flow.FlowId }));
    }
    catch (Exception ex)
    {
        TempData["TypeMsg"] = "error";
        TempData["Msg"] = ex.Message; //TODO: correct message and view
        return View(flow);
    }
}
```

Figura 7 Acción para guardar los cambios del formulario de edición del proceso

Formulario de respuesta (tarea, pregunta)

Un formulario de respuesta requiere dos acciones:

- **Mostrar el formulario de respuesta:** se ejecuta cuando el usuario ingresa a la tarea para responderla. Devuelve la vista correspondiente a ese formulario. La Figura 8 tiene un ejemplo.
- **Responder:** se ejecuta cuando el usuario hace clic en el botón que responde la tarea. La Figura 9 tiene un ejemplo.

```
[HttpGet]
public ActionResult DisplayTutorialTask(Guid flowId, Guid taskId, Guid taskToId)
{
    TaskModel model = GetTask(flowId, taskId, taskToId);
    return View( model);
}
```

Figura 8 Acción para mostrar el formulario de respuesta

```
[HttpPost]
[ValidateInput(false)] //Needed for rich text box
public ActionResult DisplayTutorialTask([ModelBinder(typeof(TaskModelBinder))]
TaskModel task)
{
    try
    {
        RespondTask(task);
        TempData["TypeMsg"] = "success";
        TempData["Msg"] = "Tarea respondida correctamente";
        return Redirect(Url.Action("Display", new { area = "", flowId =
task.FlowId }));
    }
    catch (Exception ex)
    {
        TempData["TypeMsg"] = "error";
        TempData["Msg"] = ex.Message; //TODO: correct message and view
        return View(task);
    }
}
```

Figura 9 Acción para responder la tarea

Modificación de _ViewStart.cshtml

Como ya se mencionó, Q-flow provee dos *layouts* para utilizar en los formularios. El primero, *CustomFormsLayout*, muestra el menú y el cabezal del sitio. El segundo, *CustomFormsEmptyLayout*, no los muestra. Éste es particularmente útil cuando los usuarios acceden al sitio desde los mensajes de correo electrónico que reciben de Q-flow (por ejemplo, cuando Q-flow les notifica que se espera que respondan una tarea).

Se recomienda hacer la modificación que se muestra en la Figura 10 para que los formularios utilicen el *layout* adecuado en cada caso.

```
@{  
    if (Request.QueryString["useEmptyLayout"] == null ||  
        Request.QueryString["useEmptyLayout"] != "True")  
    {  
        Layout = "~/Views/Shared/_CustomFormsLayout.cshtml";  
    }  
    else  
    {  
        Layout = "~/Views/Shared/_CustomFormsEmptyLayout.cshtml";  
    }  
}
```

Figura 10 Modificación de _ViewStart.cshtml

Compilar los archivos y publicarlos en el sitio web

Una vez que esté listo el código de los formularios personalizados, compile el proyecto. Después, copie el componente generado (por ejemplo, *SampleCustomForm.dll*) en la carpeta *Areas*, dentro del sitio de Q-flow. Copie las vistas y otros archivos cuyo contenido sea compilado en una subcarpeta de *Areas* que tenga el mismo nombre que el componente y que el área (*SampleCustomForm*, por ejemplo).

Para que el área sea considerada en el sitio de Q-flow, es necesario reiniciar el sitio en el IIS.

ASOCIAR LOS FORMULARIOS A LA PLANTILLA

Finalmente, debe asociar los formularios a la plantilla en el Diseñador Web de Procesos del Negocio. En esta herramienta, los formularios personalizados son un elemento que se puede agregar un paquete, plantilla o versión, de la misma forma que se pueden agregar datos de aplicación, roles, integraciones, etc (consulte el manual del Diseñador Web de Procesos del Negocio).

Agregar los formularios personalizados

Para agregar un formulario personalizado:

1. Abra el Diseñador Web de Procesos del Negocio.
2. Busque, en el explorador de soluciones, la plantilla del proceso al que quiere asociar los formularios. Haga clic con el botón derecho en el paquete, plantilla o versión al que desee agregarlo, y seleccione en el menú contextual la opción “Ítems”, y la subopción “Formularios personalizados”. Recuerde que los elementos de un paquete están disponibles para todos los descendientes de ese paquete, es decir, para todas las plantillas de ese paquete y todas las versiones de esa plantilla. Lo mismo pasa con los elementos de una plantilla con respecto a sus versiones. Por lo tanto, si desea que el formulario esté disponible para varias plantillas de un paquete, agréguelo al paquete. Si el formulario es específico de una plantilla, pero es el mismo para todas las versiones, agréguelo a la plantilla. El único caso en el que debe agregarlo a una versión es si es específico de esa versión.
3. Q-flow muestra el listado de formularios personalizados definidos. Para agregar uno nuevo, haga clic en el botón +. Esto abrirá un formulario que le permitirá configurar el formulario personalizado (Figura 12). Para detalles de la configuración, vea la sección “Configuración de los formularios personalizados”.

Repita el procedimiento para todos los formularios que desee.

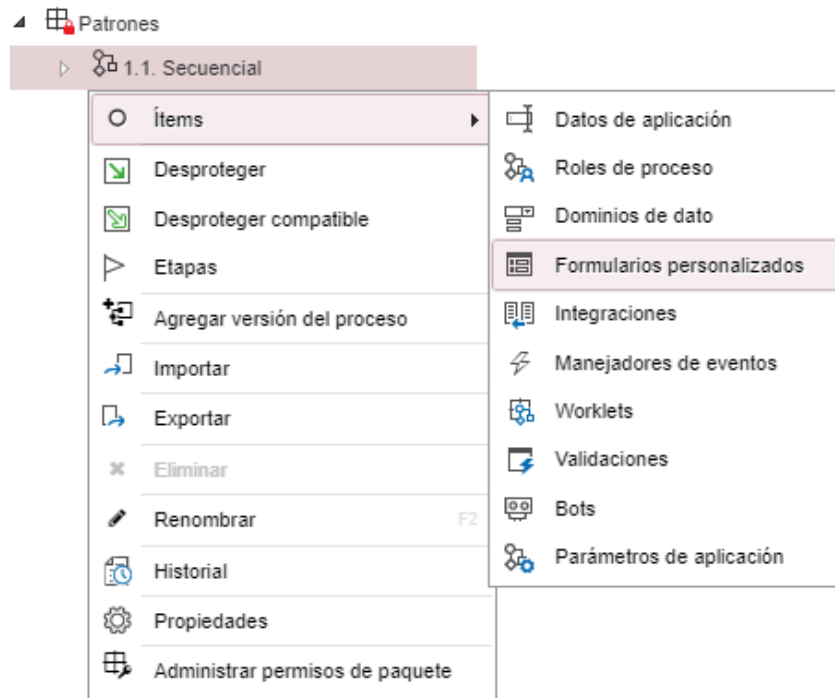


Figura 11 Agregar un formulario personalizado


The image shows a dialog box for adding a custom form. It has a dark title bar with a close button and a checkmark. Below the title bar, there is a 'General' tab. The 'Descripción' field is empty. The 'Usar MVC' checkbox is checked, and the 'Es área' checkbox is unchecked. The 'Vista' field is empty.

Figura 12 Agregar un formulario personalizado

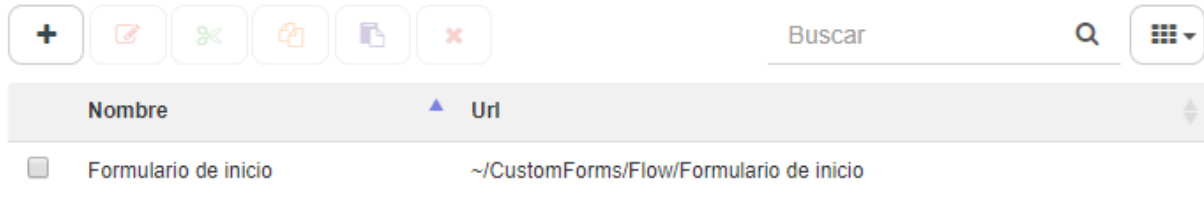
Configuración de los formularios personalizados

Una vez agregados los formularios personalizados, debe configurarlos, indicando para cada uno el archivo *cshtml* que lo contiene (si es una vista personalizada) o el área, controlador y acción que le corresponden (si los formularios fueron construidos mediante un área personalizada).

Para configurar un formulario personalizado:

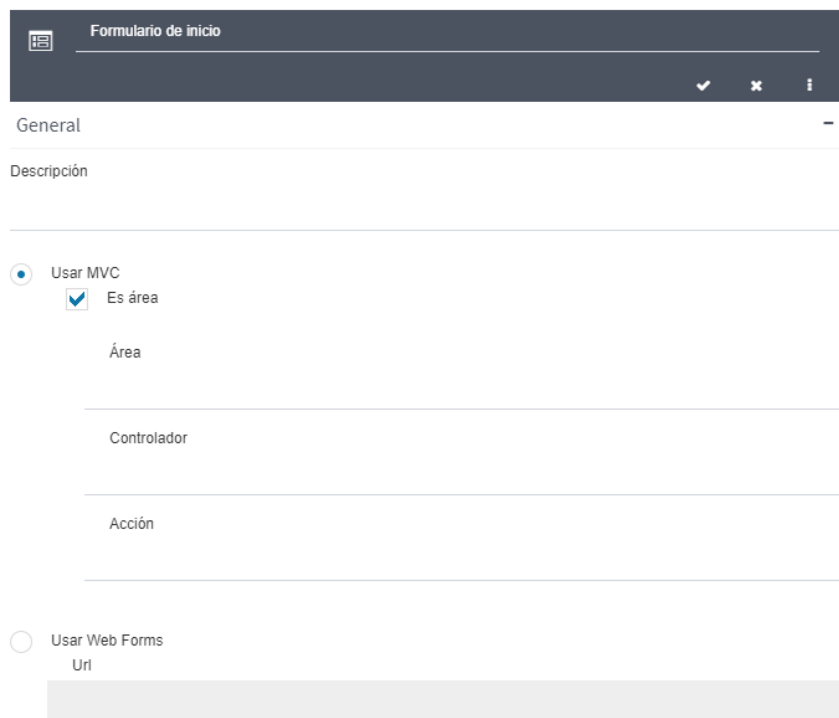
1. Abra la lista de formularios del paquete, plantilla o versión al que pertenece el formulario personalizado (Figura 13).
2. Haga doble clic en el formulario personalizado para abrir su formulario de propiedades.
3. Deje marcada la opción “Usar MVC”. La opción “Usar WebForms”, solo se visualizará cuando la opción del Web.config “ShowWebFormsProperties” tenga el valor “true” y será accesible al expandir el formulario mediante el botón , es para formularios hechos para el sitio alternativo de Q-flow, que se mantiene por compatibilidad con formularios hechos en versiones anteriores y que son diferentes a los que se describen en este manual.
4. Llene los datos del formulario.
 - a. Si el formulario es una vista personalizada, deje desmarcada la opción “Es área”, e introduzca, donde dice “Vista”, el nombre de la vista en la que se encuentra el formulario.
 - b. Si el formulario pertenece a un área, marque la opción “Es área”. La pantalla cambiará para mostrar otras propiedades, que debe llenar como se describe a continuación:
 - i. **Área:** escriba el nombre del área. Éste es el nombre que devuelve la propiedad `AreaName` de la clase que registra el área. En el ejemplo que aparece en la sección “Registrar el área” de este manual, el nombre del área es “SampleCustomForm”.
 - ii. **Controlador:** escriba el nombre de la clase que contiene el controlador, pero sin el sufijo “Controller”. Por ejemplo, si la clase es “FlowController”, escriba “Flow”.
 - iii. **Acción:** escriba el nombre de la acción correspondiente al formulario. En el ejemplo que aparece en la sección “Formulario de inicio de proceso” de este manual, todas las acciones del formulario de inicio se llaman “StartTutorial”. Tenga en cuenta que, aunque para un formulario de inicio haya tres acciones, todas ellas tienen el mismo nombre y sólo debe definir un formulario personalizado en el Diseñador Web de Procesos del Negocio.

c. Haga clic en “Aceptar” () para guardar los cambios.



Nombre	Url
<input type="checkbox"/> Formulario de inicio	~/CustomForms/Flow/Formulario de inicio

Figura 13 Formularios personalizados



Formulario de inicio

General

Descripción

☒ Usar MVC

☒ Es área

Área

Controlador

Acción

☐ Usar Web Forms

Url

Figura 14 Propiedades expandidas de un formulario personalizado

Formulario de inicio

General

Descripción

☒ Usar MVC

☒ Es área

Área

SampleCustomForm

Controlador

Flow

Acción

StartTutorial

☐ Usar Web Forms

Url

Figura 15 Configuración de un formulario personalizado que está en un área

Asociar los formularios a los elementos correspondientes

Finalmente, hay que asociar los formularios a los elementos a los que corresponden. Por ejemplo, un formulario de respuesta se asocia a un paso interactivo, como un paso de tarea o de pregunta. A qué elemento se asocia el formulario depende del tipo del formulario:

- Formulario de inicio: se asocia al paso de inicio de la versión.
- Formulario del proceso: se asocia a la versión.
- Formulario de edición del proceso: se asocia a la versión.
- Formulario de respuesta: se asocia al paso de tarea o pregunta correspondiente.

Formulario personalizado de inicio y de respuesta

El formulario personalizado de inicio va a asociado al paso de inicio de la versión en la que se lo desea utilizar. Un formulario de respuesta va asociado al paso de tarea o pregunta para el cual se lo diseñó.

Para asociar uno de estos formularios:

1. Si la versión en la que desea utilizar el formulario no está desprotegida, desprotégala (haga clic con el botón derecho en la versión y en el menú contextual, seleccione “Desproteger”).
2. Háglele doble clic a la versión en el explorador de soluciones del Diseñador Web de Procesos del Negocio, para abrir el diseño del proceso.
3. Una vez abierto el diseño, haga doble clic en el paso al cual desee asociar el formulario. Expanda el formulario y luego expanda la sección “Formulario personalizado”.
4. Seleccione el formulario que desee asociar al paso (Figura 16).
5. Proteja la versión. Recuerde que para que el formulario aparezca en el sitio web al iniciar un proceso, la versión debe ser la versión en producción y tiene que haber sido protegida después de hacer el cambio.

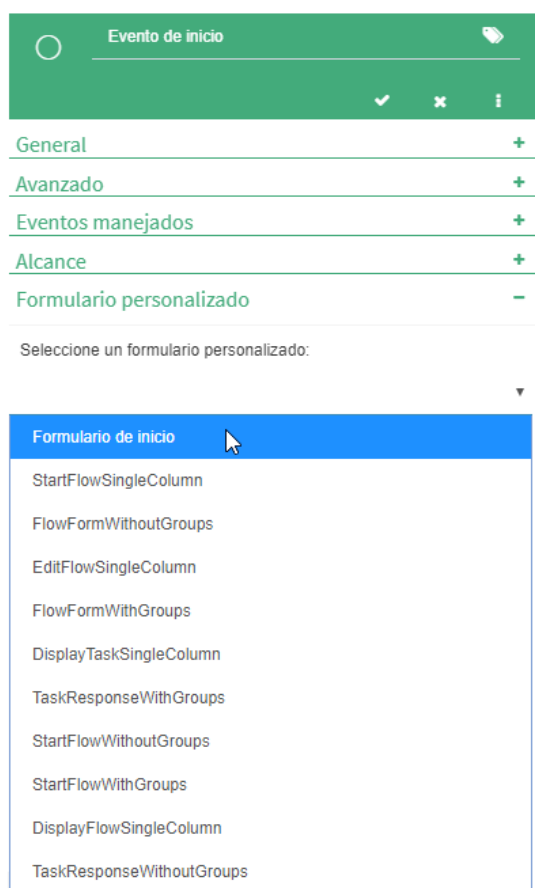


Figura 16 Selección de formulario personalizado

Formulario del proceso y Formulario de edición del proceso

Tanto el formulario del proceso como el formulario de edición del proceso van asociados a una versión. Para asociar uno de estos formularios:

1. Si la versión en la que desea utilizar el formulario no está desprotegida, desprotégala (haga clic derecho en la versión y en el menú contextual, seleccione “Desproteger”).
2. Haga clic con el botón derecho en la versión en la cual desea utilizar el formulario, y seleccione en el menú contextual “Formularios del proceso” y la opción “Formulario del proceso” o la opción “Formulario de edición del proceso”, según cuál sea el formulario que desea asociar. Q-flow abre un formulario que le permite configurar las propiedades del mismo, expanda la sección “Formulario personalizado” y luego seleccione el formulario (Figura 17).
3. Seleccione el formulario y haga clic en “Aceptar”.

Alcance	Formulario personalizado
Seleccione un formulario personalizado:	
< Ninguno >	
Validaciones	

Figura 17 Selección de formulario personalizado

Recuerde proteger la versión para poder visualizar los cambios en el sitio de Q-flow.

REFERENCIA DE HELPERS

A continuación se describe cada uno de los *helpers* que Q-flow pone a disposición de los desarrolladores para generar el código HTML de varios elementos que pueden aparecer en formularios personalizados.

- **QAttachmentTable:** permite agregar al formulario la tabla que muestra los archivos adjuntos del proceso.

- **QAttachmentUpload:** permite agregar al formulario las opciones de adjuntar archivos. Tener en cuenta que estas opciones no funcionarán correctamente si no hay permisos de adjuntar archivos en el formulario. Un formulario genérico debería verificar que estos permisos están disponibles antes de agregar estos controles (ejemplo en Figura 18). El código de este *helper* debe estar fuera del elemento “form”, a diferencia de QAttachmentTable, que debe estar dentro.
- **QControl:** permite agregar al formulario el control que se usa para mostrar un dato de aplicación y para modificar su valor (una caja de texto, una lista desplegable, etc). La etiqueta del dato de aplicación (la que le indica al usuario de qué dato se trata) se agrega por separado, mediante el *helper* QLabel. La Figura 19 muestra un ejemplo en el que se agrega el control para el dato “Nombre del cliente”.
- **QCustomValidations:** permite agregar al formulario las validaciones que se definieron para el proceso en el Diseñador Web de Procesos del Negocio.
- **QEditFlowButton:** se usa en formularios de edición del proceso. Permite agregar fácilmente el botón que guarda los cambios introducidos en el formulario.
- **QFlowInfo:** permite agregar al formulario una sección con los datos del proceso.
- **QGroup:** permite agregar al formulario los datos de aplicación de un determinado grupo. Si se omite el nombre del grupo (Figura 20), agrega los datos de aplicación que no está en ningún grupo.
- **QLabel:** permite agregar al formulario la etiqueta de un dato de aplicación (la que le indica al usuario el nombre del dato). Es necesaria siempre que se quiera mostrar el nombre del dato. Se utiliza de la misma forma que QControl.
- **QLine:** permite agregar al formulario un bloque de líneas. Los bloques de líneas se muestran como tablas y se adaptan a pantallas pequeñas.
- **QPageScrollMenu:** permite agregar al formulario un control para navegar entre las distintas secciones del formulario (Figura 21).
- **QRespondTaskButton:** se usa en formularios de respuesta. Permite agregar fácilmente el botón que responde la tarea.
- **QRole:** funciona de forma análoga a QControl, pero para un rol.
- **QRoles:** permite agregar al formulario los roles disponibles. Si el alcance de un rol para ese formulario determina que es editable, el rol se muestra como editable.
- **QStartFlowButton:** se usa en formularios de inicio. Permite agregar fácilmente el botón que inicia el proceso.
- **QTaskInfo:** permite agregar al formulario una sección con los datos de la tarea.
- **QTemplateInfo:** permite agregar al formulario una sección con los datos de la plantilla.
- **QValue:** permite agregar al formulario una etiqueta que muestra el valor de un dato de aplicación (no permite modificar el dato de aplicación).

```
@if (Model.AttachmentsModel.Scope.AddAllowed)
{
    @Html.QAttachmentUpload(Model.AttachmentsModel, new
    QSimplePanelOptions())
}
```

Figura 18 Antes de agregar el control para agregar adjuntos, se verifica que en el formulario haya permisos para agregarlos

```
@Html.QControl(Model.Data["Nombre del cliente"]);
```

Figura 19 Uso de QControl para agregar el dato de aplicación "Nombre del cliente".

```
@Html.QGroup("", new QGroupPanelOptions() { GroupOptions = new
QSimplePanelOptions() { StyleClass = "box box-danger", AddParentDiv = true,
ParentDivClass = "col-lg-12" } })
```

Figura 20 QGroup: el primer parámetro es el nombre del grupo



Figura 21 El control para navegar entre las secciones del formulario, que se puede agregar mediante el *helper* QPageScrollMenu

REFERENCIA DE LA BIBLIOTECA JAVASCRIPT

Q-flow provee una biblioteca de JavaScript con funciones para ser utilizadas desde formularios personalizados. Las funciones son accesibles por medio del objeto *Host*. El objeto *Host* permite obtener otros objetos, como por ejemplo, objetos que representan datos de aplicación, que

también exponen funciones y propiedades. Todas las funciones que tienen índices (típicamente, el parámetro *instance*) utilizan el 0 como primera posición.

Host

El objeto *Host* expone las siguientes funciones. Para acceder a ellas, escriba *Host.nombreFunción(parámetros)*. Tenga en cuenta que algunas funciones sólo tienen sentido en algunos contextos. Por ejemplo, no tiene sentido *getTaskName* en un formulario de inicio, porque no hay tarea.

- **getData(dataName):** devuelve un objeto *Data*, que representa el dato de aplicación cuyo nombre es *dataName*.
- **getLine(lineName):** devuelve un objeto *Line* que representa el bloque de líneas cuyo nombre es *lineName*.
- **getRole(roleName):** devuelve un objeto *Role* que representa el rol cuyo nombre es *roleName*.
- **getDataSourceItemDescription(key, domainId, parameters, onSuccess, onError):** para utilizar con dominios que obtienen sus datos de fuentes externas de datos (o de una lista; ver el manual Diseñador Web de Procesos del Negocio). Obtiene, para el dominio con el identificador *domainId*, la descripción correspondiente a la clave *key*. Si el dominio recibe parámetros, estos se pasan en el parámetro *parameters*. Los parámetros *onSuccess* y *onError* son para especificar funciones a invocar en caso de éxito o error. La función especificada en *onSuccess* puede recibir un parámetro, que Q-flow llenará con la descripción obtenida.
- **getSecurityMember(memberId, onSuccess, onError):** obtiene el miembro (usuario, cola de trabajo, nodo, etc) correspondiente al identificador *memberId*. Si lo encuentra con éxito, invoca la función especificada por *onSuccess*, pasando como parámetro el miembro obtenido. Si falla, invoca la función especificada en *onError*.
- **getFlowName:** devuelve el nombre del proceso.
- **getFlowDescription:** devuelve la descripción del proceso.
- **getTemplateName:** devuelve el nombre de la plantilla a la que pertenece el proceso.
- **getTemplateDescription:** devuelve la descripción de la plantilla a la que pertenece el proceso.
- **getTemplateVersionName:** devuelve el nombre de la versión que se usó para iniciar el proceso.
- **getTemplateVersionDescription:** devuelve la descripción de la versión que se usó para iniciar el proceso.

- **getTaskName:** devuelve el nombre de la tarea.
- **getTaskDescription:** devuelve la descripción de la tarea.
- **getTaskSubject:** devuelve el asunto de la tarea.
- **getTaskResponse:** devuelve la respuesta a la tarea.
- **getTaskProgress:** devuelve el progreso de la tarea.
- **setFlowName(value):** asigna *value* al nombre del proceso (usar sólo en un formulario de inicio).
- **setFlowDescription(value):** asigna *value* a la descripción del proceso (usar sólo en un formulario de inicio).
- **setTaskResponse(value):** selecciona la respuesta con valor *value*.
- **setTaskProgress(value):** asigna *value* al progreso de la tarea.
- **submit():** envía el formulario (es como hacer clic en el botón principal del formulario).

Data

Un objeto *Data* representa un dato de aplicación. Los objetos *Data* se obtienen mediante la función *Host.getData*. Exponen las siguientes funciones:

- **addInstance():** en datos multivaluados, agrega un valor al dato, sin especificar ese valor. El valor debe ser asignado después mediante la función *setValue*.
- **getInstanceCount():** devuelve la cantidad de valores que tiene el dato.
- **getValue(instance):** devuelve el valor que tiene el dato en la posición indicada por *instance* (por ejemplo, *getValue(0)* devuelve el primer valor).
- **removeInstance(instance):** elimina el valor en la posición indicada por *instance*.
- **setErrorMessage(itemInstance, msg):** muestra el mensaje de error especificado por el parámetro *msg* para el dato en la posición indicada por el parámetro *itemInstance*.
- **setValue(instance, value):** asigna el valor *value* a la posición *instance* del dato. Cabe destacar que se asume que el valor es del tipo de dato del dominio configurado.

Las siguientes propiedades de los objetos *Data* pueden resultar útiles:

- **addAllowed:** indica si el alcance definido para el dato en el formulario permite agregar valores al dato.
- **dataId:** identificador del dato.
- **dataType:** representa tipo del dato.
- **defaultValue:** valor por defecto del dato.
- **domainId:** identificador del dominio del dato.

- **group:** nombre del grupo al que pertenece el dato.
- **isMultivalued:** indica si el dato es multivaluado.
- **lineName:** nombre del bloque de línea.
- **maxInstances:** cantidad máxima de valores que puede tener el dato.
- **minInstances:** cantidad mínima de valores que debe tener el dato.
- **name:** nombre del dato.
- **removeAllowed:** indica si el alcance definido para el dato en el formulario permite quitar valores del dato.
- **scope:** representa el alcance del dato para el formulario en el que se está usando.
- **tooltip:** *tooltip* del dato.

Line

Un objeto *Line* representa un bloque de líneas. Los objetos *Line* se obtienen mediante la función *Host.getLine*. Exponen las siguientes funciones:

- **addInstance():** agrega una fila al bloque.
- **getData(dataName):** devuelve el dato cuyo nombre se indica en el parámetro *dataName*. El objeto devuelto es del tipo *Data*. Un bloque de líneas está compuesto por varios datos, uno por cada columna del bloque, y todos multivaluados. Cada valor de cada dato representa una celda del bloque de líneas. Por ejemplo, para obtener el valor que hay en la segunda fila de un bloque de líneas para determinada columna, hay que buscar el segundo valor del dato correspondiente a esa columna (si la columna está representada por el dato “Dirección”, hay que buscar el segundo valor del dato “Dirección”).
- **getInstanceCount():** devuelve la cantidad de filas que tiene el bloque.
- **removeInstance(instance):** elimina la fila en la posición indicada por *instance*.

Las siguientes propiedades de los objetos *Line* pueden resultar útiles:

- **addAllowed:** indica si el alcance del bloque para ese formulario permite agregar filas a la línea.
- **maxLines:** cantidad máxima de filas que puede tener la línea.
- **minLines:** cantidad mínima de filas que debe tener la línea.
- **name:** nombre del bloque de líneas.
- **removeAllowed:** indica si el alcance del bloque para ese formulario permite borrar filas de la línea.

Role

Un objeto *Role* representa un rol. Los objetos *Role* se obtienen mediante la función *Host.getRole*. Exponen las siguientes funciones:

- **addInstance():** permite agregar un miembro al rol, sin especificar cuál es el miembro.
- **addMember(memberId, onSuccess, onError):** agrega al rol el miembro con identificador *memberId*. Una vez agregado con éxito el miembro, Q-flow ejecutará la función indicada en *onSuccess* (esta función puede recibir como parámetro un objeto que representa el miembro seleccionado). Si se produce un error, invocará la función especificada en *onError*.
- **getInstanceCount():** devuelve la cantidad de miembros que tiene el rol.
- **getMemberId(instance):** devuelve el identificador del miembro que está en la posición *instance*.
- **getMemberName(instance):** devuelve el nombre del miembro que está en la posición *instance*.
- **getMemberType(instance):** devuelve el tipo del miembro (por ejemplo, si es un usuario, una cola de trabajo, etc) que está en la posición indicada por *instance*.
- **removeInstance(instance):** quita del rol el miembro que está en la posición *instance*.
- **removeMember(instance):** quita del rol el miembro en la posición *instance*.
- **setErrorMessage(itemInstance, msg):** muestra el mensaje de error especificado por el parámetro *msg* para el miembro en la posición indicada por el parámetro *itemInstance*.
- **setMember(value, instance, onSuccess, onError):** asigna el miembro indicado por *value* a la posición *instance* del rol. Una vez agregado con éxito el miembro, Q-flow ejecutará la función indicada en *onSuccess* (esta función puede recibir como parámetro un objeto que representa el miembro seleccionado). Si se produce un error, invocará la función especificada en *onError*.

Las siguientes propiedades de los objetos *Role* pueden resultar útiles:

- **addAllowed:** indica si el alcance definido para el rol en el formulario permite agregar miembros al dato.
- **id:** identificador del rol.
- **isMultivalued:** indica si el rol es multivaluado.
- **maxInstances:** cantidad máxima de valores que puede tener el rol.
- **minInstances:** cantidad mínima de valores que debe tener el rol.
- **name:** nombre del rol.

- **removeAllowed:** indica si el alcance definido para el rol en el formulario permite quitar miembros del rol.