

Q-flow 3.0: Web services

Código del manual: Qf303008ESP
Versión: 1.0
Se aplica a: Q-flow 3.03
Última revisión: 29/9/2008

Q-flow 3.0

Web services

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

Tabla de Contenido

Introducción	4
Organización de este manual	4
Convenciones usadas en este manual	4
Descripción de los web services	4
WebStart	4
Métodos de WebStart	4
Ejemplos de uso de WebStart	5
WebResponse	7
Métodos de WebResponse	7
Ejemplos de uso de WebResponse	8
WebLists	9
Métodos de WebLists	9
Ejemplos de uso de WebLists	9
WebLinks	10
Método de WebLinks	10
WebOperations	10
Métodos de WebOperations	10

Introducción

El propósito de este manual es describir los web services de Q-flow para apoyar a desarrolladores de aplicaciones que los consumen.

Este manual contiene solamente información acerca de la interfaz y del comportamiento de los web services. Para obtener información acerca de cómo instalarlos y configurarlos, consulte el manual de instalación de Q-flow.

Organización de este manual

Q-flow incluye los siguientes web services:

- WebStart
- WebResponse
- WebLists
- WebLinks
- WebOperations

El manual dedica una sección a cada uno de esos web services.

Convenciones usadas en este manual

Este manual representa la firma de un método de un web service de la siguiente forma:

Nombre_del_método(lista_de_parámetros):Tipo_de_Returno

Los parámetros están separados por comas, y cada parámetro está representado por su tipo seguido de su nombre, como en código C#. Ejemplo: string name. Si un método no retorna nada, el tipo de retorno es "void", como en C#.

El código de los ejemplos está escrito en el lenguaje C#.

Descripción de los web services

A continuación, se describe cada uno de los web services de Q-flow.

WebStart

El web service WebStart tiene métodos que inician workflows.

Métodos de WebStart

- **StartFlowNow(Guid templateId, string title, string description):Guid** – inicia un workflow basado en el template indicado por el parámetro templateId. El parámetro title indica el título

que se le dará al workflow, y el parámetro description indica la descripción. Devuelve el identificador global (Guid) del workflow iniciado.

- **StartFlowNowCorrelativeID(long templateCorrelativeID, string title, string description):Guid** – hace lo mismo que el método anterior, pero en lugar de recibir un parámetro del tipo Guid para identificar el template en el que se debe basar el nuevo workflow, recibe un parámetro de tipo long, que indica el identificador correlativo del template a ser utilizado.
- **StartFlow(FlowMessage flowMessage):Guid** – hace lo mismo que los métodos anteriores, pero recibe un parámetro del tipo FlowMessage para indicar los datos del workflow a ser iniciado. El objeto FlowMessage, además de tener propiedades que representan el título y la descripción del workflow, tiene propiedades que representan los datos de aplicación, roles y archivos adjuntos del workflow. Estas propiedades pueden ser modificadas para inicializar los datos, roles y archivos adjuntos del workflow a iniciar. Para obtener el objeto FlowMessage a utilizar para invocar este método, primero debe invocar el método GetFlowInfoCorrelativeId o el método GetNewFlowInfo.
- **GetNewFlowInfoCorrelativeId(long templateCorrelativeID):FlowMessage** – dado un template indicado por su identificador correlativo, devuelve un objeto de tipo FlowMessage que puede ser utilizado para iniciar un workflow con el método StartFlow.
- **GetNewFlowInfo(Guid templateId):FlowMessage** – dado un template indicado por su identificador global, devuelve un objeto de tipo FlowMessage que puede ser utilizado para iniciar in workflow con el método StartFlow.

Ejemplos de uso de WebStart

Esta sección contiene algunos ejemplos de cómo usar el web service WebStart.

Ejemplo que utiliza StartFlowNowCorrelativeID

Este ejemplo supone que existe un template cuyo identificador correlativo es 1202.

```
WebStart.WebStart wsStart = new WebStart.WebStart();
Guid flowId;
flowId = wsStart.StartFlowNowCorrelativeID(1202, "WS Test", "Test");
```

Ejemplo que utiliza StartFlow

Este ejemplo supone que existe un template cuyo identificador correlativo es 1202. Utiliza el método GetNewFlowInfoCorrelativeId para obtener un objeto del tipo FlowMessage. Una vez obtenido ese objeto, se le asigna el título del workflow (FlowName), la descripción del workflow (FlowDescription) y se invoca el método StartFlow pasándole por parámetro el objeto de tipo FlowMessage obtenido y modificado. El resultado es el mismo que el del método anterior.

```
WebStart.FlowMessage flowMsg;
WebStart.WebStart wsStart = new WebStart.WebStart();
flowMsg = wsStart.GetNewFlowInfoCorrelativeId(1202);
flowMsg.FlowName = "WS Test";
flowMsg.FlowDescription = "Test";
Guid flowId;
flowId = wsStart.StartFlow(flowMsg);
```

Ejemplo con inicialización de datos y roles

Este ejemplo obtiene un objeto del tipo `FlowMessage` por medio de una invocación al método `GetNewFlowInfoCorrelativeId`. Después, inicializa el dato "amount" con un valor recibido en un parámetro, y el rol "Approver" con un valor recibido en otro parámetro. Finalmente, define un nombre y una descripción para el workflow, y lo inicia por medio de una invocación a `StartFlow`, pasando el objeto de tipo `FlowMessage` como parámetro.

```
private Guid StartExpenseApprovalFlow(string amount, Guid approver)
{
    WebStart.WebStart wsStart = new WebStart.WebStart();
    WebStart.FlowMessage flowMsg =
    wsStart.GetNewFlowInfoCorrelativeId(1202);
    foreach (WebStart.DataMessage datum in flowMsg.Data)
    {
        if (datum.Name == "amount")
        {
            if (datum.Scope == WebStart.ItemScope.Editable)
                datum.Values[0].Value = amount;
            else
                throw new Exception("El dato debería ser
editable.");
        }
    }
    foreach (WebStart.RoleMessage role in flowMsg.Roles)
    {
        if (role.Name == "Approver")
        {
            if (role.Scope == WebStart.ItemScope.Editable)
            {
                role.Members[0].Type =
                WebStart.TemplateRoleMemberType.User;
                role.Members[0].ID = approver;
            }
            else
                throw new Exception("El rol debería ser
editable.");
        }
    }
    flowMsg.FlowName = "WS Test";
    flowMsg.FlowDescription = "Test";
    Guid flowId = wsStart.StartFlow(flowMsg);
    return flowId;
}
```

Ejemplo de inicio de workflow con un archivo adjunto

Este ejemplo inicia un workflow, adjuntándole un archivo que está en el directorio raíz y que se llama "ejercicios.pdf". El paso de inicio del template en el que se basa el workflow debe tener el alcance configurado para que se pueda agregar archivos adjuntos. De lo contrario, el ejemplo produce un error.

```

private void StartFlowWithAttachment(long templateCorrelativeId)
{
    WebStart.WebStart wsStart = new WebStart.WebStart();
    WebStart.FlowMessage flowMsg =
    wsStart.GetNewFlowInfoCorrelativeId(templateCorrelativeId);
    flowMsg.FlowName = "WS Test";
    flowMsg.FlowDescription = "Test";
    flowMsg.Attachments = CreateAttachments();
    wsStart.StartFlow(flowMsg);
}

private WebStart.AttachmentMessageBase[] CreateAttachments()
{
    WebStart.AttachmentMessage file = new WebStart.AttachmentMessage();
    file.Content = File.ReadAllBytes("C:\\ejercicios.pdf");
    WebStart.AttachmentMessageBase[] attachments = new
    WebStart.AttachmentMessageBase[1];
    attachments[0] = file;
    attachments[0].Name = "ejercicios.pdf";
    return attachments;
}

```

WebResponse

El web service WebResponse tiene métodos para responder tareas.

Métodos de WebResponse

- **GetTask(Guid FlowId, Guid StepId, Guid Told):TaskMessage** – obtiene un objeto que representa la tarea determinada por el identificador de un workflow (FlowId), un paso de ese workflow (StepId) y un destinatario de ese paso (Told). Una vez obtenido ese objeto, es posible usarlo para responder la tarea que representa por medio de alguno de los otros métodos de ese web service.
- **RespondTask(TaskMessage task):void** – responde la tarea representada por parámetro. Ese objeto debe haber sido obtenido por medio del método GetTask y ser modificado para indicar la respuesta que se da a la tarea y, opcionalmente, un porcentaje de progreso.
- **RespondTaskNow(Guid FlowId, Guid StepId, Guid Told, string ResponseKey):void** – responde una tarea con la respuesta indicada por el parámetro ResponseKey. La tarea que se responde está determinada por el identificador de un workflow (FlowId), un paso de ese workflow (StepId) y un destinatario de ese paso (Told).
- **RespondTaskNowWithProgress(Guid FlowId, Guid StepId, Guid Told, string ResponseKey, byte Progress):void** – responde una tarea con la respuesta indicada por el parámetro ResponseKey, y actualizando el progreso de la tarea con el valor del parámetro Progress. La tarea que se responde está determinada por el identificador de un workflow (FlowId), un paso de ese workflow (StepId) y un destinatario de ese paso (Told).

Ejemplos de uso de WebResponse

Esta sección contiene algunos ejemplos de cómo usar el web service WebResponse.

Ejemplo que utiliza RespondTaskNow

Este ejemplo primero utiliza el web service WebLists para obtener las tareas pendientes del usuario actual. Después, utiliza el web service WebResponse para responder la primera de esas tareas.

El ejemplo supone que la tarea que contesta admite la respuesta "Approve". Si una tarea es contestada utilizando una respuesta que no está prevista, ocurre un error.

```
WebLists.WebLists wsLists = new WebLists.WebLists();
WebLists.Task[] tasks = wsLists.GetCurrentUserTasks();
Guid flowId = tasks[0].FlowID;
Guid stepId = tasks[0].TaskID;
Guid toId = tasks[0].ToID;
WebResponse.WebResponse wsResponse = new WebResponse.WebResponse();
wsResponse.RespondTaskNow(flowId, stepId, toId, "Approve");
```

Ejemplo que utiliza RespondTask

Este ejemplo utiliza el web service WebLists para obtener las tareas pendientes del usuario actual. Después, utiliza los datos de la primera tarea obtenida para invocar el web method GetTask, que devuelve un objeto del tipo TaskMessage. Una vez obtenido ese objeto, le carga la respuesta "Approved" y lo utiliza para invocar el método RespondTask. Eso responde la tarea con la respuesta "Approve".

El ejemplo supone que la tarea que contesta admite la respuesta "Approve". Si una tarea es contestada utilizando una respuesta que no está prevista, ocurre un error.

```
WebLists.WebLists wsLists = new WebLists.WebLists();
WebLists.Task[] tasks = wsLists.GetCurrentUserTasks();
Guid flowId = tasks[0].FlowID;
Guid stepId = tasks[0].TaskID;
Guid toId = tasks[0].ToID;
WebResponse.WebResponse wsResponse = new WebResponse.WebResponse();
WebResponse.TaskMessage taskMsg;
taskMsg = wsResponse.GetTask(flowId, stepId, toId);
taskMsg.ResponseKey = "Approve";
wsResponse.RespondTask(taskMsg);
```

En este caso, la tarea que se contesta corresponde a un paso de pregunta. Si se tratara de un paso de tarea, se podría actualizar el progreso, escribiendo una línea como la siguiente:

```
taskMsg.Progress = 50;
```

Ejemplo de respuesta que modifica datos de aplicación

Este ejemplo hace lo mismo que el ejemplo anterior, salvo por el hecho de que modifica un dato de aplicación del workflow al que pertenece la tarea que es contestada.

El ejemplo supone que la tarea que contesta admite la respuesta "Approve". Si una tarea es contestada utilizando una respuesta que no está prevista, ocurre un error.

```
private void RespondTask(string comentarios)
{
    WebLists.WebLists wsLists = new WebLists.WebLists();
    WebLists.Task[] tasks = wsLists.GetCurrentUserTasks();
    Guid flowId = tasks[0].FlowID;
    Guid stepId = tasks[0].TaskID;
    Guid toId = tasks[0].ToID;
    WebResponse.WebResponse wsResponse = new WebResponse.WebResponse();
    WebResponse.TaskMessage taskMsg = wsResponse.GetTask(flowId,
stepId, toId);
    taskMsg.ResponseKey = "Approve";
    foreach (WebResponse.DataMessage datum in taskMsg.Data)
    {
        if (datum.Name == "comentarios")
        {
            if (datum.Scope == WebResponse.ItemScope.Editable)
                datum.Values[0].Value = comentarios;
            else
                throw new Exception("El dato debería ser
editable.");
        }
    }
    wsResponse.RespondTask(taskMsg);
}
```

WebLists

El web service web lists tiene métodos para obtener listas de tareas.

Métodos de WebLists

- **GetCurrentUserTasks():Task[]**– obtiene la lista de tareas del usuario en cuyo contexto se invoca el método.
- **GetFlowTasks(Guid flowid):Task[]** – obtiene la lista de tareas del workflow cuyo identificador es indicado por el parámetro flowid.

Ejemplos de uso de WebLists

Esta sección contiene un ejemplo de cómo usar el web service WebLists.

Ejemplo que usa `GetCurrentUserTasks` y `GetFlowTasks`

Este ejemplo utiliza el método `GetCurrentTasks` para obtener las tareas pendientes del usuario actual. Después, utiliza `GetFlowTasks` para obtener todas las tareas del workflow al que pertenece la primera de las tareas obtenidas anteriormente.

```
WebLists.WebLists wsLists = new WebLists.WebLists();
WebLists.Task[] tasks = wsLists.GetCurrentUserTasks();
Guid flowId = tasks[0].FlowID;
tasks = wsLists.GetFlowTasks(flowId);
```

WebLinks

El web service `WebLinks` dispone solamente de un método.

Método de `WebLinks`

- **`HasUserPermissionsToSeeLink(Guid linkId):bool`**: devuelve "true" si el usuario actual (aquél cuyas credenciales se utilizan para invocar el web service) tiene permiso para ver el vínculo cuyo identificador es el valor del parámetro `linkId`.

WebOperations

El web service `WebOperations` trabaja en conjunto con el paso de sincronización (ver manual del diseñador de procesos del negocio). Un paso de sincronización puede ser configurado para que espere la ocurrencia de una acción externa. La forma de indicarle a Q-flow que la acción externa tuvo lugar es invocar alguno de los métodos del web service `WebOperations`.

Métodos de `WebOperations`

- **`FinalizeWaitingStepsByName(Guid flowId, string templateStepName): void`**: dado un paso de sincronización, especificado por el identificador del workflow al que pertenece y el nombre del paso, indica que ese paso debe terminar la espera por una acción externa.
- **`FinalizeWaitingStepsById(Guid flowId, Guid templateStepId): void`**: dado un paso de sincronización, especificado por el identificador del workflow al que pertenece y el identificador del paso del template en el que se basa, indica que ese paso debe terminar la espera por una acción externa.