

Referencia de la Interfaz de Scripting

21/09/2018

Q-flow 4.2

Tabla de Contenido

Introducción	3
Referencia de la interfaz de scripting de Q-flow.....	3
Scripts de pasos de código, de manejadores de eventos y de integraciones	4
CodeScriptBase	4
Scripts de pasos de evaluación por código	5
CodeEvaluationScriptBase	6
Clases manejadas por los scripts	6
Attachment.....	6
Data	7
DataInstance.....	8
DBConnectionConfiguration	9
FlowStage	9
FlowContext.....	10
IScriptHost	11
Parameter	16
Role.....	16
RoleMember.....	17
Task.....	18
TaskTo.....	18
TemplateParameter.....	18
ThreadContext	20
User	20
WSConnectionConfiguration	21

INTRODUCCIÓN

Los pasos de código, manejadores de eventos y otros elementos de Q-flow utilizan scripts. Q-flow incluye un conjunto de clases que permiten manipular información de Q-flow dentro de esos scripts. Dichas clases están en el namespace Qflow.Steps.Configuration.Scripting. Este manual describe esas clases.

Este manual también le es útil a quienes quieran desarrollar un componente para ser utilizado en una integración de tipo Qflow Assembly, puesto que esos componentes definen clases derivadas de la clase CodeScriptBase de Q-flow, y esta clase es descrita en este manual.

REFERENCIA DE LA INTERFAZ DE SCRIPTING DE Q-FLOW

Los scripts deben ser escritos en C# o Visual Basic .Net, y pueden acceder a las clases de cualquiera de los namespaces incluidos en los siguientes componentes:

- System.dll
- System.Xml.dll
- System.Data.dll
- System.Data.OracleClient.dll
- System.Core.dll
- System.Data.DataSetExtensions.dll
- System.Xml.Linq.dll

Para utilizar clases de otros componentes debe utilizar una integración de tipo “Assembly” o “Qflow Assembly”. Consulte el manual del diseñador de procesos del negocio para obtener más información sobre esos tipos de integración. Naturalmente, los componentes que sean utilizados en integraciones de esos tipos pueden haber sido desarrollados en cualquier lenguaje compatible con el framework .NET.

Scripts de pasos de código, de manejadores de eventos y de integraciones

Un script de un paso de código, de un manejador de eventos o de una integración debe declarar una clase derivada de la clase `CodeScriptBase`. Cuando un paso de código es agregado a la definición de un proceso, Q-flow automáticamente crea en el script el código que declara la clase y la relación de herencia, además del método `Execute()`. Lo mismo pasa cuando un manejador de eventos es creado. En el caso de los scripts de integraciones, Q-flow crea todo el código del script, aunque el usuario tiene la posibilidad de modificarlo si así lo desea.

El método `Execute()` es un método abstracto de la clase `CodeScriptBase`, y por lo tanto debe ser implementado en el script. Cuando Q-flow ejecuta un paso de código, crea un objeto de la clase declarada en el script e invoca el método `Execute()` de ese objeto. Lo mismo sucede cuando ejecuta una integración y cuando ejecuta el manejador de un evento.

CodeScriptBase

La clase `CodeScriptBase` es la clase base de todos los scripts de los pasos de código. Sus objetos tienen un conjunto de métodos y propiedades protegidos que, por lo tanto, también están disponibles en objetos de sus clases derivadas. Estos métodos y propiedades devuelven objetos de otras clases que también pertenecen al namespace `Qflow.Steps.Configuration.Scripting`. Estas clases se describen más adelante.

Propiedades

- **Attachments: List<Attachment>**: permite obtener la lista de archivos adjuntos del proceso. Esta lista no sirve para agregarle archivos adjuntos al proceso. Para eso se debe utilizar el método `AddAttachment` del objeto `Host`. Además, los objetos `Attachment` de esa lista contienen solamente los datos de los adjuntos; no contienen el contenido de los adjuntos. Para obtener el contenido de un adjunto, utilice el método `GetAttachmentContent` del objeto `Host`.
- **Context: IThreadContext**: representa el hilo en el que se encuentra el paso de código, y permite acceder y manipular información referente a ese hilo.
- **Data: List<Data>**: permite obtener la lista de datos de aplicación del proceso.
- **Flow: IFlowContext**: representa el proceso en el que se encuentra el paso de código, y permite acceder y manipular su información.
- **Host: IScriptHost**: devuelve un objeto de tipo `IScriptHost`, que contiene funciones que permiten obtener información general.

- **Parameters: List<Parameter>:** devuelve una lista de parámetros. Estos parámetros sólo son útiles en los scripts de las integraciones.
- **Roles: List<Roles>:** permite obtener la lista de roles del proceso.

Métodos

- **GetAttachment(Guid attachmentId): Attachment:** permite obtener el archivo adjunto correspondiente al Guid indicado.
- **GetAttachment(string attachmentName): Attachment:** permite obtener el archivo adjunto correspondiente al nombre indicado.
- **GetData(Guid guid): Data:** permite obtener el dato de aplicación correspondiente al Guid indicado.
- **GetData(string name): Data:** permite obtener un dato de aplicación cuyo nombre es el indicado en el parámetro. Si hay más de un dato de aplicación con ese nombre, devuelve el primero que encuentra.
- **GetParameterData(string paramName): Data:** este método sólo es útil en los scripts de las integraciones. Es utilizado internamente por Q-flow y permite obtener el dato de aplicación asociado a un determinado parámetro.
- **GetRole(Guid guid): Role:** permite obtener el rol correspondiente al Guid indicado.
- **GetRole(string name): Role:** permite obtener un rol cuyo nombre es el indicado en el parámetro. Si hay más de un rol con ese nombre, devuelve el primero que encuentra.
- **Initialize(IScriptHost host, ThreadContext context):** inicializa la clase con el objeto Host y el objeto ThreadContext indicados. Este método es invocado por Q-flow antes de ejecutar el script.

Scripts de pasos de evaluación por código

El script de un paso de evaluación por código debe declarar una clase derivada de la clase `CodeEvaluationScriptBase`. Cuando un paso de código es agregado a la definición de un proceso, Q-flow automáticamente crea en el script el código que declara la clase y la relación de herencia, además del método `Evaluate()`.

El método `Evaluate()` es un método abstracto de la clase `CodeEvaluationScriptBase`, y por lo tanto debe ser implementado en el script del paso de código. Cuando Q-flow ejecute el paso de código, creará un objeto de la clase declarada en el script e invocará el método `Evaluate()` de ese objeto. El resultado de esa invocación, que es “verdadero” o “falso”, determina el resultado de la evaluación del paso.

CodeEvaluationScriptBase

Las clase `CodeEvaluationScriptBase` es la clase base de todos los scripts de los pasos de evaluación por código. Tiene un conjunto de métodos y propiedades protegidos y, por lo tanto, disponibles en sus clases derivadas. Estos métodos y propiedades devuelven objetos de otras clases que también pertenecen al namespace `Qflow.Steps.Configuration.Scripting`, y que son descritas más adelante. Para ver las descripciones de las propiedades y los métodos disponibles en `CodeEvaluationScriptBase`, consulte la sección sobre la clase `CodeScriptBase`.

Clases manejadas por los scripts

Las propiedades de las clases de los scripts devuelven objetos de otras clases. Estos objetos representan procesos, hilos, datos y otros elementos. Esta sección describe esas clases.

Attachment

Los objetos de la clase `Attachment` representan archivos adjuntos al proceso, y son los objetos que Q-flow guarda en la colección `Attachments` de los objetos `FlowContext`.

Constructores

La clase `Attachment` tiene constructores que Q-flow utiliza internamente y no están pensados para ser utilizados en los scripts. Q-flow crea estos objetos automáticamente al crear los objetos de las clases de los scripts. Utilizar estos constructores no debería ser necesario durante el desarrollo de scripts, y no es recomendable.

- **`Attachment()`**: crea un objeto `Attachment` vacío.
- **`Attachment(Guid attachID)`**: crea un objeto `Attachment` con el Guid indicado.
- **`Attachment(Guid attachID, int attachVersion)`**: crea un objeto `Attachment` con el Guid y la versión indicados.
- **`Attachment(Guid attachID, int attachVersion, string name)`**: crea un objeto `Attachment` con el Guid, la versión y el nombre indicados.

Propiedades

- **`AttachID: Guid`**: permite obtener el identificador del archivo adjunto.
- **`AttachVersion: int`**: permite obtener la versión del archivo adjunto.
- **`CustomProperties: NameValueCollection`**: permite acceder a las propiedades extendidas del adjunto.
- **`Description: string`**: permite especificar u obtener la descripción del archivo adjunto.
- **`Name: string`**: permite especificar u obtener el nombre del archivo adjunto.

- **Size: long:** devuelve el tamaño en bytes del contenido del adjunto.
- **TimeCreated: DateTime:** devuelve la fecha de creación del adjunto.
- **TimeLastModified: DateTime:** devuelve la fecha en que el adjunto fue modificado por última vez.

Data

Los objetos de la clase Data representan datos de aplicación y son los objetos que Q-flow guarda en la colección Data de los objetos FlowContext.

Constructores

La clase Data tiene tres constructores que Q-flow utiliza internamente y no están pensados para ser utilizados en los scripts. Q-flow crea estos objetos automáticamente al crear los objetos de las clases de los scripts. Utilizar estos constructores no debería ser necesario durante el desarrollo de scripts, y no es recomendable.

- **Data():** crea un objeto Data vacío.
- **Data(Guid dataID, string name, Type dataType, bool isArray):** crea un objeto Data con los parámetros indicados.
- **Data(Guid dataID, string name, Type dataType, bool isArray, string lineBlock):** similar al anterior, pero permite especificar un bloque de línea para el dato.

Propiedades

- **DataID: Guid:** permite obtener el identificador del dato.
- **DataType: Type:** permite obtener el tipo del dato.
- **DomainID: Guid:** permite obtener el identificador del dominio al que pertenece el dato.
- **GroupName: string:** permite obtener el nombre del grupo de datos al que pertenece el dato.
- **isArray: bool:** indica si el dato es multivaluado o no.
- **IsNullValue: bool:** indica si el valor del dato es nulo. Los datos pueden tener valores nulos, pero la propiedad Value siempre devuelve un valor (si el dato es nulo, devuelve el valor por defecto del dominio del dato). Por lo tanto, para saber si un dato es nulo, se debe utilizar esta propiedad.
- **LineBlock: string:** permite averiguar a qué bloque de líneas pertenece el dato.
- **Name: string:** permite obtener el nombre del dato.
- **Value: object:** permite obtener el valor del dato. Si el dato tiene múltiples valores, devuelve el primero. El objeto que se devuelve tiene el mismo tipo del dato (string,

bool, decimal, DateTime o Guid), aunque se devuelva por medio de una referencia de tipo Object para que la propiedad pueda devolver objetos de varios tipos distintos.

- **Values: List<DataInstance>:** en el caso de datos con valores múltiples o pertenecientes a un bloque de línea, permite especificar u obtener la lista de valores del dato. Esta propiedad tiene un indexador que permite acceder a uno de los valores a través de su índice por medio de una notación abreviada. Por ejemplo, si hay una variable de tipo Data llamada “datos”, se puede acceder al tercer valor por medio de la notación “datos[2]”.

Métodos

- **AddValue(object value):** agrega un valor al dato (si el dato pertenece a un bloque de líneas, agrega una línea).
- **AddValue(object value, int instanceIndex):** agrega al dato un valor en la posición indicada por el parámetro instanceIndex (si el dato pertenece a un bloque de líneas, agrega una línea)
- **Clear():** vacía la lista de valores del dato.
- **DelValue(int instance):** borra el dato de la posición indicada por el parámetro “instance”.

DataInstance

Los objetos de la clase DataInstance representan valores de un dato que tiene múltiples valores o de un dato que pertenece a un bloque de líneas. La clase DataInstance es la clase de los objetos que son guardados en la colección Values de los objetos de la clase Data.

Constructores

Los constructores de la clase DataInstance pueden ser utilizados en los scripts para crear nuevas instancias y agregarlas como valores a los datos de aplicación.

- **DataInstance():** crea un objeto DataInstance, sin cargarle nada.
- **DataInstance(Data data, long instance, object value):** crea un objeto DataInstance asignándole el dato indicado por el parámetro “data”, el número de instancia indicado por el parámetro “instance” y el valor indicado por el parámetro “value”. Tenga en cuenta que no agrega el objeto a la lista de valores de ningún dato.

Propiedades

- **Data: Data:** permite especificar u obtener el dato al que pertenece el DataInstance.

- **Instance: long:** permite especificar u obtener la posición que tiene el DataInstance en la colección de valores del dato al que pertenece.
- **IsNull: bool:** indica si el valor es nulo. Los datos pueden tener valores nulos, pero la propiedad Value siempre devuelve un valor (si el dato es nulo, devuelve el valor por defecto del dominio del dato). Por lo tanto, para saber si un dato es nulo, se debe utilizar esta propiedad.
- **Value: object:** permite especificar u obtener el valor del DataInstance.

DBConnectionConfiguration

Los objetos de la clase DBConnectionConfiguration representan parámetros de aplicación que almacenan datos de conexión a bases de datos.

Constructores

- **DBConnectionConfiguration()**
- **DBConnectionConfiguration(string connString, string provider, string dataBase):** crea un objeto con los datos indicados (cadena de conexión, proveedor de base de datos y nombre de la base de datos respectivamente).

Propiedades

- **ConnectionString: string:** cadena de conexión a una base de datos.
- **DataBase: string:** nombre de la base de datos.
- **Provider: string:** nombre del proveedor de base de datos. Ejemplo: System.Data.SqlClient.

FlowStage

Esta clase incluye las propiedades básicas de una etapa, y se utiliza solamente para proveer información de la etapa actual del flow.

Propiedades

- **Stageld: Guid:** identificador de la etapa.
- **Name: string:** nombre de la etapa.
- **Status: StageStaus:** estado de cumplimiento de la etapa. Es un enumerado con alguno de estos valores: SlackTime, OnTime, Delayed.

FlowContext

La clase FlowContext permite manipular información del proceso que está ejecutando el script. La propiedad Flow disponible en los scripts devuelve un objeto de este tipo.

Constructores

La clase FlowContext tiene dos constructores que Q-flow utiliza internamente y no están pensados para ser utilizados en los scripts. Q-flow crea estos objetos automáticamente al crear los objetos de las clases de los scripts. Utilizar estos constructores no debería ser necesario durante el desarrollo de scripts, y no es recomendable.

- **FlowContext():** crea un objeto FlowContext vacío.
- **FlowContext(Guid flowId, long flowCorrelativeId, Guid templateId, Guid templateVersionId):** crea un objeto FlowContext, asignándole el identificador del proceso, el identificador correlativo del proceso, el identificador de la plantilla de proceso y el identificador de la versión indicados. No carga información del proceso cuyo identificador se indica.

Propiedades

- **Attachments: List<Attachment>:** permite obtener una lista que contiene los archivos adjuntos al proceso. Esta lista sirve para obtener los archivos adjuntos de un proceso, pero no para agregarle archivos adjuntos.
- **CurrentStage: FlowStage:** propiedad que tiene información de la etapa actual del flow.
- **Data: List<Data>:** permite obtener una lista que contiene los datos de aplicación del proceso.
- **Description: string:** permite especificar u obtener la descripción del proceso.
- **Flag: string:** permite especificar u obtener la bandera del proceso. La bandera es lo que se coloca en los pasos de hito y en muchos otros pasos. Es una etiqueta que describe el estado del proceso con un texto cualquiera.
- **FlowCorrelativeID: long:** devuelve el identificador correlativo del proceso. El identificador correlativo es un número que es asignado al proceso en el momento de su creación. Es secuencial, es decir, el número de un proceso es igual al número del anterior más uno.
- **FlowID: Guid:** devuelve el identificador del proceso.
- **Importance: byte:** permite especificar u obtener la importancia del proceso. La importancia indica la prioridad.
- **Name: string:** permite especificar u obtener el nombre del proceso.
- **Progress: byte:** permite especificar u obtener el progreso del proceso.
- **Roles: List<Roles>:** permite obtener una lista que contiene los roles del proceso.

- **StartDate: DateTime:** fecha de inicio del proceso.
- **StarterUserID: Guid:** identificador del usuario que inició el proceso.
- **TemplateID: Guid:** devuelve el identificador de la plantilla en el que se basa el proceso.
- **VersionID: Guid:** devuelve el identificador de la versión en la que se basa el proceso.

Métodos

- **GetAttachment(Guid attachmentId): Attachment:** permite obtener un objeto de tipo Attachment que representa el archivo adjunto cuyo identificador se pasó por parámetro.
- **GetAttachment(string attachmentName): Attachment:** permite obtener un objeto de tipo Attachment que representa el archivo adjunto cuyo nombre se pasó por parámetro.
- **GetData(Guid guid): Data:** permite obtener el dato de aplicación correspondiente al Guid indicado.
- **GetData(string name): Data:** permite obtener un dato de aplicación cuyo nombre es el indicado en el parámetro. Si hay más de un dato de aplicación con ese nombre, devuelve el primero que encuentra.
- **GetRole(Guid guid): Role:** permite obtener el rol correspondiente al Guid indicado.
- **GetRole(string name): Role:** permite obtener un rol cuyo nombre es el indicado en el parámetro. Si hay más de un rol con ese nombre, devuelve el primero que encuentra.

IScriptHost

Los objetos de la interfaz IScriptHost expone los siguientes métodos:

- **AddAttachment(string name, byte[] content):** agrega al proceso un archivo adjunto. El parámetro *name* especifica el nombre que tendrá el nuevo adjunto, y el parámetro *content* es el contenido binario de ese archivo. La función maneja versiones de los adjuntos: si ya existe un adjunto con el nombre del adjunto ingresado, se crea una nueva versión del adjunto ya existente.
- **AddAttachment(string path):** agrega al proceso como archivo adjunto el archivo cuya ruta es indicada por el parámetro *path*.
- **AddSubstitution(Guid userID, DateTime from, DateTime to, Guid substituteID):** crea una suplencia para el usuario correspondiente al identificador *userID*, desde la fecha especificada por el parámetro *from* hasta la fecha indicada por el parámetro *to*, con el usuario correspondiente al identificador *substituteID* como suplente.
- **FinalizeWaitingStep(Guid templateStepId): void:** este método permite señalarle al proceso que un paso de sincronización debe terminar su espera y que se debe continuar su ejecución. El paso de sincronización tiene que estar configurado para que

espere una acción externa. La invocación de este método es la “acción externa” que indica que la espera debe terminar. El valor del parámetro *templateStepId* es el identificador del paso de sincronización cuya espera se desea finalizar.

- **FinalizeWaitingStep(string templateStepName): void:** este método se comporta de la misma forma que el anterior, pero en lugar de recibir por parámetro el identificador del paso de sincronización cuya espera se desea finalizar, recibe el nombre de ese paso.
- **GetAttachmentContent(string name): byte[]:** dado el nombre de un archivo adjunto, devuelve el contenido de ese archivo en la forma de una secuencia de bytes.
- **GetCalendarDate(Guid calendarID, DateTime date): DateTime:** dado un día (*date*), si ese día es un día de trabajo según el calendario indicado por el parámetro *calendarID*, devuelve ese mismo día. Si ese día no es un día de trabajo, devuelve el primer día de trabajo posterior a ese día. Por ejemplo: si el día indicado en el parámetro *date* es un sábado, y el calendario especifica que se trabaja de lunes a viernes, la función devuelve un objeto que representa el primer lunes posterior a ese sábado. También toma en cuenta las horas. Por ejemplo: si la hora de salida es las 18:00, y el parámetro *date* indica jueves a las 19:00, el método devuelve el viernes siguiente a ese jueves.
- **GetCalendarDate(Guid calendarID, DateTime date, TimeSpan span): DateTime:** dado un día (*date*) y un período de tiempo (*span*), le suma a ese día ese período de tiempo. Si el resultado es un día de trabajo según el calendario de trabajo, devuelve ese mismo día. De lo contrario, devuelve el primer día de trabajo posterior a ese día. También toma en cuenta el horario de trabajo, por lo que, por ejemplo, si el resultado es un DateTime que indica una hora posterior a la jornada laboral, devuelve el día siguiente con la hora de inicio de la jornada.
- **GetCalendarDate(Guid calendarID, DateTime date, int span): DateTime:** dado un día (*date*) y una cantidad de días (*span*), le suma a ese día la cantidad de días. Si el resultado es un día de trabajo según el calendario, devuelve ese mismo día. De lo contrario, devuelve el primer día de trabajo posterior a ese día. A diferencia de la anterior, no tiene en cuenta el horario de trabajo. Aunque el resultado sea un DateTime que indica una hora posterior a la jornada laboral, devuelve el mismo día que el indicado en el resultado, y no el día siguiente.
- **GetCustomParameter(string parameterKey): string:** devuelve el valor del parámetro personalizado correspondiente a la clave *parameterKey*.
- **GetDataSourceItemDescription(Guid dataDomainId, string keyValue): string:** dado el identificador de un dominio de datos y la clave de uno de sus registros, devuelve el valor correspondiente a esa clave.
- **GetDataSourceItemDescription(Guid dataDomainId, string keyValue, NameValueCollection parameters): string:** similar al anterior, pero recibe además una

colección de pares (nombre, valor) para especificar los valores de los parámetros del dominio. Cada elemento de la colección corresponde a un parámetro del dominio, con su nombre y valor.

- **GetDirectManagersOfGroup(Guid groupId): Guid[]:** devuelve un *array* con los identificadores de los usuarios que son supervisores de un grupo de forma directa. Si A es supervisor del grupo G, y el grupo H es miembro del grupo G, A no es supervisor de forma directa del grupo H, pero sí del grupo G. Es, sin embargo, supervisor de forma indirecta del grupo H. El parámetro *groupId* puede ser también el identificador de un nodo.
- **GetFlowRoleMembers(Guid templateRoleId): Guid[]:** devuelve un *array* con los identificadores de los usuarios que están desempeñando el rol identificado por *templateRoleId* en el proceso actual.
- **GetManagedUsers(Guid userID): Guid[]:** devuelve en un *array* de objetos Guid los identificadores de los usuarios supervisados por el usuario cuyo identificador coincide con el valor del parámetro *userID*.
- **GetManagersOf(Guid userID): Guid[]:** devuelve en un *array* de objetos Guid los identificadores de los supervisores del usuario cuyo identificador coincide con el valor del parámetro *userID*.
- **GetSecurityMemberID(string memberName, MemberType memberType): Guid:** obtiene el identificador de un usuario, grupo o nodo, a partir de su nombre (*memberName*) y tipo (indicado por el parámetro *memberType*).
- **GetSystemParameter(string propertyKey): string:** obtiene el valor del parámetro de instalación indicado por el parámetro *propertyKey*.
- **GetTask(Guid flowStepID): Task:** devuelve la tarea que corresponde al paso cuyo identificador coincide con el valor del parámetro *flowStepID*.
- **GetTask(string flowStepName): Task:** devuelve la tarea que corresponde al paso cuyo nombre coincide con el valor del parámetro *flowStepName*. Si hay dos pasos con el mismo nombre, devuelve el que fue creado más recientemente. Esto puede pasar si el diseño del proceso tiene un bucle: puede haber varios pasos del proceso que correspondan al mismo paso en la definición. También puede pasar si dos pasos del diseño del proceso directamente tienen el mismo nombre.
- **GetTemplateParameter(Guid parameterId): TemplateParameter:** devuelve el parámetro de aplicación que tenga el identificador indicado.
- **GetTemplateParameter(string parameterName): TemplateParameter** devuelve el parámetro de aplicación que tenga el nombre indicado.
- **GetUser(Guid userID): User:** dado el identificador de un usuario, devuelve un objeto de tipo User que representa al usuario que corresponde a ese identificador.

- **GetUser(string loginName): User:** dado un texto que indica el proveedor de seguridad y el logon (nombre de usuario) de un usuario, devuelve un objeto User que representa a ese usuario. El proveedor de seguridad y el nombre de usuario deben ser provistos en el formato “proveedor@logon” o en el formato “proveedor\logon” (en este último caso, si el código está en C#, hay que escribir “proveedor\\logon”, dado que “\” es el carácter de escape).
- **GetUser(string loginName, string securityProviderName): User:** dado el nombre de logon de un usuario (*logonName*) y el nombre de su proveedor de seguridad (*securityProviderName*) devuelve un objeto de tipo User que representa a ese usuario.
- **GetUserID(string logonName): Guid:** dado un texto que indica el proveedor de seguridad y el logon (nombre de usuario) de un usuario, devuelve el identificador de ese usuario. El proveedor de seguridad y el nombre de usuario deben ser provistos en el formato “proveedor@logon” o en el format “proveedor\logon” (en este último caso, si el código está en C#, hay que escribir “proveedor\\logon”, dado que “\” es el carácter de escape).
- **GetUserID(string logonName, string securityProviderName): Guid:** dado el nombre de logon de un usuario (*logonName*) y el nombre de su proveedor de seguridad (*securityProviderName*) devuelve el identificador de ese usuario.
- **GetUsersByExtendedProperty(string propertyName, int value): Guid[]:** devuelve los identificadores de aquellos usuarios para los que la propiedad indicada por el parámetro *propertyName* tiene el valor indicado por el parámetro *value*.
- **GetUsersByExtendedProperty(string propertyName, string value): Guid[]:** devuelve los identificadores de aquellos usuarios para los que la propiedad indicada por el parámetro *propertyName* tiene el valor indicado por el parámetro *value*.
- **GetUsersByMemberID(Guid memberID): Guid[]:** dado el identificador de algún miembro del organigrama (usuario, nodo o grupo), devuelve los identificadores de los usuarios que pertenecen a ese miembro y de los usuarios que pertenecen a los descendientes de ese miembro. Por ejemplo, si el identificador corresponde a un grupo, devuelve todos los usuarios de ese grupo, todos los usuarios de los grupos que pertenecen a ese grupo, y así sucesivamente. Si el identificador corresponde a un usuario, devuelve el identificador de ese usuario.
- **GetUsersTaskLoad(Guid[] users): Dictionary<Guid, int>:** dado un conjunto de usuarios especificados por sus identificadores, devuelve un diccionario cuya clave es el identificador de cada usuario y cuyo valor es la cantidad de tareas que ese usuario tiene asignadas.
- **GetUsersTaskLoadForTemplate(Guid[] users): Dictionary<Guid, int>:** dado un conjunto de usuarios especificados por sus identificadores, devuelve un diccionario cuya clave es el identificador de cada usuario y cuyo valor es la cantidad de tareas que

- ese usuario tiene asignadas en procesos con la misma plantilla que el que ejecuta el script.
- **GetUserWithLeastTasks(Guid[] users): Guid:** dado un conjunto de usuarios, indicados éstos por sus identificadores, devuelve el identificador de aquél de esos usuarios que tenga menos tareas.
 - **GetUserWithLeastTasksForTemplate(Guid[] users): Guid:** dado un conjunto de usuarios, indicados éstos por sus identificadores, devuelve el identificador de aquél de esos usuarios que tenga menos tareas en procesos con la misma plantilla que el que ejecuta el script.
 - **ResetFlowCounter(Guid flowStepID):** dado el identificador de un paso “Repetir”, pone en 0 el contador que cuenta la cantidad de iteraciones realizadas por ese paso. Es útil para casos en los que ocurre un error durante una iteración y se quiere volver a reintentar la ejecución del paso como si éste nunca se hubiera ejecutado.
 - **ResetFlowCounter(string flowStepName):** dado el nombre de un paso “Repetir”, pone en 0 el contador que cuenta la cantidad de iteraciones realizadas por ese paso. Es útil para casos en los que ocurre un error durante una iteración y se quiere volver a reintentar la ejecución del paso como si éste nunca se hubiera ejecutado.
 - **ResolveAddressees(Guid templateStepID): Guid[]:** dado el identificador de un paso de la definición de un proceso, devuelve un array con los identificadores de los usuarios a los que ese paso está destinado, sustituyendo los roles por los identificadores de los usuarios que los desempeñan.
 - **ResolveAddressees(string templateStepName): Guid[]:** dado el nombre de un paso de la definición de un proceso, devuelve un array con los identificadores de los usuarios a los que ese paso está destinado, sustituyendo los roles por los identificadores de los usuarios que los desempeñan.
 - **RespondTask(Guid flowStepID, string responseKey):** responde la tarea cuyo identificador coincide con el valor del parámetro *flowStepID*. El valor de la respuesta es el valor del parámetro *responseKey*. La tarea queda registrada como contestada por el primero de sus destinatarios.
 - **RespondTask(Guid flowStepID, string responseKey, byte progress):** responde la tarea cuyo identificador coincide con el valor del parámetro *flowStepID*. El valor de la respuesta es el valor del parámetro *responseKey*. La tarea queda registrada como contestada por el primero de sus destinatarios. El porcentaje de progreso de la tarea queda actualizado con el valor del parámetro *progress*.
 - **RespondTask(string flowStepName, string responseKey):** responde la tarea correspondiente al paso cuyo nombre coincide con el valor de *flowStepName*. Si hay más de un paso en esas condiciones, toma en cuenta el que fue iniciado más

recientemente. El valor de la respuesta es el valor del parámetro *responseKey*. La tarea queda registrada como contestada por el primero de sus destinatarios.

- **RespondTask(string flowStepName, string responseKey, byte progress):** responde la tarea correspondiente al paso cuyo nombre coincide con el valor de *flowStepName*. Si hay más de un paso en esas condiciones, toma en cuenta el que fue iniciado más recientemente. El valor de la respuesta es el valor del parámetro *responseKey*. La tarea queda registrada como contestada por el primero de sus destinatarios. El porcentaje de progreso de la tarea queda actualizado con el valor del parámetro *progress*.

Parameter

Los objetos de la clase *Parameter* son utilizados internamente por Q-flow en los scripts de integraciones. Son los objetos que se guardan en la lista *Parameters* de los objetos *ThreadContext*. Representan los parámetros de las integraciones y su asociación con datos de aplicación del proceso. La clase *Parameter* fue desarrollada sólo para uso interno de Q-flow, y no se recomienda utilizarla. No tiene, además, aplicación fuera del ámbito de Q-flow, por lo que utilizarla en un script es absolutamente innecesario.

Constructor

- **Parameter(string name, Data data):** crea un parámetro con el nombre indicado por el argumento *name* y asociado al dato representado por el argumento *data*.

Propiedades

- **Name: string:** permite obtener el nombre del parámetro.
- **Data: Data:** permite especificar u obtener el objeto que representa el dato asociado al parámetro.

Role

Los objetos de la clase *Role* representan roles de los procesos y contienen la información que indica que usuario o usuarios los están desempeñando. Son los objetos que Q-flow guarda en la colección *Roles* de los objetos *FlowContext*.

Constructores

Los constructores de la clase *Role* son utilizados internamente por Q-flow y no están pensados para ser utilizados en los scripts. Q-flow crea estos objetos automáticamente al crear los objetos de las clases de los scripts. Utilizar estos constructores no debería ser necesario durante el desarrollo de scripts, y no es recomendable.

- **Role():** crea un rol vacío.

- **Role(Guid roleId, string roleName):** crea un rol con el identificador y nombre indicados.
- **Role(Guid roleId, string roleName, bool isMultiUser):** crea un rol con el identificador y el nombre indicados. El parámetro *isMultiUser* indica si el rol debe admitir múltiples miembros o no.

Propiedades

- **IsMultiUser: bool:** indica si el rol es multivaluado (si puede tener más de un miembro).
- **Member: RoleMember:** permite obtener el miembro del rol. Si el rol tiene muchos miembros, devuelve el primer miembro del rol.
- **Members: List<RoleMember>:** permite especificar u obtener la lista de miembros del rol.
- **Name: string:** devuelve el nombre del rol.
- **RoleId: Guid:** devuelve el identificador del rol.

RoleMember

Los objetos de la clase RoleMember representan miembros de roles. Indican, para un rol, qué usuario lo desempeña. Son los objetos que Q-flow guarda en la colección Members de los objetos de la clase Role.

Constructores

Los constructores de la clase RoleMember pueden ser utilizados en los scripts para crear nuevas instancias y agregarlas como miembros de roles de proceso.

- **RoleMember():** crea un miembro de rol vacío.
- **RoleMember(Guid memberId):** crea un miembro de rol con el identificador indicado.
- **RoleMember(Guid memberId, string name):** crea un miembro de rol con el identificador y el nombre indicados.
- **RoleMember(Guid memberId, string name, MemberType memberType):** crea un miembro de rol con el identificador, el nombre y el tipo especificados.

Propiedades

- **MemberID: Guid:** permite obtener el identificador del miembro de rol.
- **MemberType: MemberType:** permite obtener el tipo del rol.
- **Name: string:** permite obtener el nombre del miembro de rol.

Task

La clase Task permite manipular información de una tarea. Los métodos de nombre GetTask de IScriptHost devuelven un objeto de este tipo.

Propiedades

- **Description: string:** permite obtener la descripción de la tarea.
- **Name: string:** permite obtener el nombre de la tarea.
- **StepID: Guid:** permite obtener el identificador del paso del proceso correspondiente a la tarea.
- **TaskTo: List<TaskTo>:** permite obtener la lista de tareas individuales ("tareas a") de la tarea. Cada paso de tarea tiene una tarea individual por cada uno de sus destinatarios.
- **TimeEnded: DateTime?:** permite obtener la fecha en la que terminó la tarea. Devuelve "null" si la tarea todavía no terminó.
- **TimeStarted: DateTime:** permite obtener la fecha en la que se inició la tarea.

TaskTo

La clase TaskTo permite manipular información de una tarea dirigida (una tarea está compuesta por una o más tareas individuales; una por destinatario de la tarea).

Propiedades

- **DateRead: DateTime?:** fecha en la que se leyó la tarea.
- **Progress: byte:** permite obtener el porcentaje de progreso de la tarea.
- **ReadByUserID: Guid:** permite obtener el identificador del usuario que leyó la tarea.
- **Responded: DateTime?:** permite obtener la fecha en la que el destinatario respondió la tarea. Devuelve null si aún no hay respuesta.
- **RespondedByUserID: Guid:** devuelve el identificador del usuario que contestó la tarea.
- **ResponseKey: string:** devuelve la clave de la respuesta dada.
- **StepToID: Guid:** devuelve el identificador de la tarea individual.
- **Subject: string:** devuelve el asunto de la tarea dirigida.
- **UserID: Guid:** devuelve el identificador del destinatario de la tarea individual.

TemplateParameter

Los objetos de la clase TemplateParameter representan parámetros de aplicación.

Propiedades

- **Id: Guid:** identificador del parámetro de aplicación.

- **Name: string:** nombre del parámetro de aplicación.
- **ParameterType: TemplateParameterType:** tipo del parámetro de aplicación (conexión a base de datos, conexión a web service, contraseña o texto).

Métodos

- **CreateWSProxyAssembly(): Assembly:** este método sólo tiene sentido si el parámetro de aplicación guarda información de conexión a un web service. El método genera un proxy para trabajar con el web service, y devuelve un objeto que representa el ensamblado .NET que contiene el código del proxy.
- **GetDBConnection(): DBConnection:** este método sólo tiene sentido si el parámetro de aplicación guarda información de conexión a una base de datos. El método devuelve un objeto del tipo System.Data.DbConnection que representa una conexión a la base de datos y que se puede utilizar para trabajar con ella por medio de los otros objetos del namespace System.Data.
- **GetDBConnectionConfiguration(): DBConnectionConfiguration:** este método sólo tiene sentido si el parámetro de aplicación guarda información de conexión a una base de datos. El método devuelve un objeto que contiene los datos de conexión a la base de datos (cadena de conexión, proveedor y nombre de la base de datos).
- **GetWSConnectionConfiguration(): WSConnectionConfiguration:** este método sólo tiene sentido si el parámetro de aplicación guarda información de conexión a un web service. El método devuelve un objeto que contiene los datos de conexión al web service.
- **GetPassword(): string:** Descifra la contraseña guardada y la devuelve. Este método sólo tiene sentido para parámetros del tipo "Contraseña".
- **GetText(): string:** este método sólo tiene sentido si el parámetro de aplicación es del tipo "Texto". Devuelve el texto guardado.
- **GetWSProxyInstance(Type type): object:** este método sólo tiene sentido si el parámetro de aplicación guarda información de conexión a un web service. Devuelve un proxy que permite trabajar con el web service cuyos datos de conexión están guardados en el parámetro de aplicación. El parámetro "type" es un objeto que representa la clase del proxy.
GetWSProxyInstance(): object: este método sólo tiene sentido si el parámetro de aplicación guarda información de conexión a un web service. Devuelve un proxy que permite trabajar con el web service cuyos datos de conexión están guardados en el parámetro de aplicación.

ThreadContext

La clase ThreadContext permite manipular información del hilo que está ejecutando el script. La propiedad Thread disponible en los scripts devuelve un objeto de este tipo.

Constructores

La clase ThreadContext tiene dos constructores que Q-flow utiliza internamente y no están pensados para ser utilizados en los scripts. Q-flow crea estos objetos automáticamente al crear los objetos de las clases de los scripts. Utilizar estos constructores no debería ser necesario durante el desarrollo de scripts.

- **ThreadContext(Guid threadID, Guid flowStepID, FlowContext flow):** crea un objeto ThreadContext con el Guid, el Guid de paso y el objeto FlowContext indicados. No carga información del hilo.
- **ThreadContext(Guid threadID, Guid flowStepID, FlowContext flow, List<Parameter> parameters):** crea un objeto ThreadContext de la misma forma que el anterior, pero además permite especificar una lista de parámetros. No carga información del hilo.

Propiedades

- **ErrorLog: string:** si el hilo está en error, permite obtener el mensaje del error.
- **Flow: FlowContext:** permite obtener un objeto que representa el proceso al que pertenece el hilo.
- **FlowStepID: Guid:** permite obtener el Guid del paso actual del hilo.
- **Parameters: List<Parameter>:** permite especificar u obtener una lista de parámetros. Los parámetros son utilizados en los scripts de integraciones, pero no son útiles en otros scripts.
- **ThreadID: Guid:** permite obtener el Guid del hilo.

Métodos

- **GetParameter(string paramName): Parameter:** devuelve el parámetro con el nombre especificado.
- **GetParameterData(string paramName): Data:** devuelve el dato de aplicación correspondiente al parámetro cuyo nombre se indica.

User

La clase User representa un usuario de Q-flow.

Propiedades

- **CalendarID: Guid:** devuelve el identificador del calendario del usuario.

- **Culture: CultureInfo:** cultura (datos de idioma) del usuario.
- **DomainName: string:** devuelve el nombre del dominio al que pertenece la cuenta del usuario.
- **Email: string:** devuelve la dirección de correo electrónico del usuario.
- **ExtendedProperties: NameValueCollection:** devuelve la lista de propiedades personalizadas del usuario (propiedades que no vienen con Q-flow y que fueron creadas por la organización; ver manual del administrador del modelo organizacional y manual de instalación).
- **IsEnabled: bool:** indica el usuario está habilitado.
- **LoginName: string:** devuelve el login (nombre de la cuenta) del usuario.
- **Name: string:** devuelve el nombre del usuario.
- **NodeID: Guid:** devuelve el identificador del nodo al que pertenece el usuario.
- **SecurityProviderName: string:** devuelve el nombre del proveedor de seguridad del usuario.
- **UserID: Guid:** devuelve el identificador del usuario.

WSConnectionConfiguration

Los objetos de la clase WSConnectionConfiguration representan parámetros de aplicación que almacenan datos de conexión a web services.

Propiedades

- **Password: string:** devuelve la contraseña correspondiente a las credenciales que se utilizan para invocar el web service si no se usan las credenciales de la red.
- **Url: string:** devuelve la URL del web service.
- **UseNetworkCredentials: bool:** indica si se debe utilizar las credenciales de la red o si se debe especificar credenciales para invocar el web service.
- **UserName: string:** nombre de usuario que se utilizará para invocar el web service si no se usan las credenciales de la red.