

Actualizado 16/11/2009



Manual SQLJOBS v 1.0.7

Módulo para DotNetNuke 4.8.0 y superiores



Javier Antó

MANUAL SQLJOBS V 1.0.7

MÓDULO PARA DOTNETNUKE 4.8.0 Y SUPERIORES

SQLJOBS es un módulo para DotNetNuke 4.8.0 y superiores con el que podrá programar la ejecución periódica de sentencias SQL o de procedimientos almacenados. El módulo hace uso del programador de tareas standard de DotNetNuke y no es necesaria ninguna modificación adicional, solo instalarlo y es completamente operativo.

Con SqlJobs podrá programar la exportación periódica a un fichero CSV y podrá enviar este fichero adjunto en un correo a cualquier dirección de E-Mail. Programe la recepción de sencillos informes SQL en su buzón de forma semanal, diaria etc...

El módulo funciona perfectamente en DotNetNuke 5.1.4

INDICE

Indice	2
Programador de tareas Dotnetnuke - Consideraciones	3
Introducción a SQLJobs	3
Primeros pasos.....	4
Configuración inicial del módulo	4
Gestión de Tareas	4
Agregue la primera Tarea	6
Ejecución de procedimientos almacenados pasando parametros y exportar los resultados.....	7
Personalización avanzada de tareas en la versión Fuente	7

PROGRAMADOR DE TAREAS DOTNETNUKE - CONSIDERACIONES

Una de las limitaciones del programador de tareas es que no puede funcionar 24/7 sin la ayuda de un programa externo. Esto es una limitación de ASP.NET, no de DotNetNuke. El proceso de trabajo "Worker Process" usado por IIS recicla de forma periódica la aplicación de acuerdo con lo especificado en el fichero "machine.config". Algunos proveedores pueden tener configuraciones que reciclan el proceso cada 30 minutos (forzado), mientras que otros pueden tener configuraciones más complicadas, como reciclar el proceso después de 3000 "hits, o tras 20 minutos de inactividad. Es éste reciclado el que apaga el programador, hasta que el proceso vuelve a arrancar de nuevo (Normalmente porque alguien visita su sitio y vuelve a arrancar la aplicación activando también el programador de tareas).

En un entorno de hosting compartido, esta funcionalidad proporciona beneficios al conjunto de las aplicaciones ya que evita la sobrecarga del servidor, dejando "dormidas" aquellas aplicaciones que no se usan, sin embargo, también es la causa de las limitaciones que podemos observar en el programador de tareas DotNetNuke.

Vemos que la limitación es que el programador funcionará 24/7, mientras alguien esté constantemente visitando el sitio web. Es durante los periodos "durmientes" que el programador deja de funcionar.

Por este motivo se ha de ser cauto a la hora de definir los tipos de tareas que el programador va a ejecutar, asegúrese de que sus tareas no tienen que ejecutarse "cada día a las 12 en punto". Tareas más apropiadas son aquellas que se han de ejecutar "Una vez por día" o "2 veces por minuto" y además no debería importarle si las tareas no se ejecutan durante periodos de inactividad.

INTRODUCCIÓN A SQLJOBS

SQLJobs hace uso del programador de tareas interno de DotNetNuke para permitir la ejecución periódica de tareas SQL, de procedimientos almacenados o, en la versión fuente, de "clases Visual Basic".

SQLJobs acepta distintas configuraciones y amplía la capacidad de ejecución del programador de tareas. El usuario puede definir la ejecución de cualquier procedimiento almacenado de la base de datos, sentencias SQL o, en la versión fuente, desviar la tarea a una clase externa para ejecutar código personalizado "Visual Basic".

Es posible definir una tarea como SQLCommand y asignar un timeout, esto es especialmente útil cuando las consultas se realizan sobre bases de datos muy grandes o son muy complejas.

Pueden exportarse los resultados de una consulta "Select" a un fichero CSV separado por comas. Este fichero estará ubicado en el servidor y además se enviará adjunto a un mensaje a las direcciones especificadas en la configuración de la tarea.

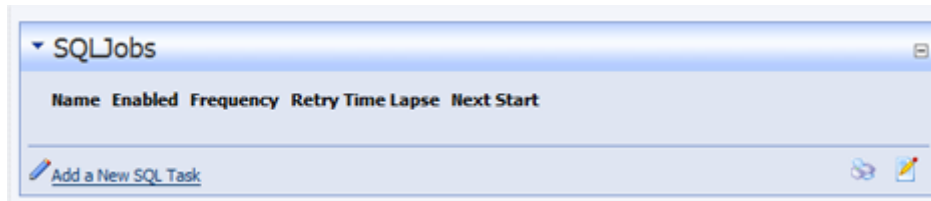
Cada vez que se ejecuta la tarea, existe la opción de enviar un mensaje de correo de aviso a varias direcciones de correo electrónico, si la casilla exportar está activada se incluye el fichero de exportación en el mensaje.

Se guarda un registro histórico de ejecuciones de la tarea junto con el resultado y resumen de registros afectados.

PRIMEROS PASOS

CONFIGURACIÓN INICIAL DEL MÓDULO

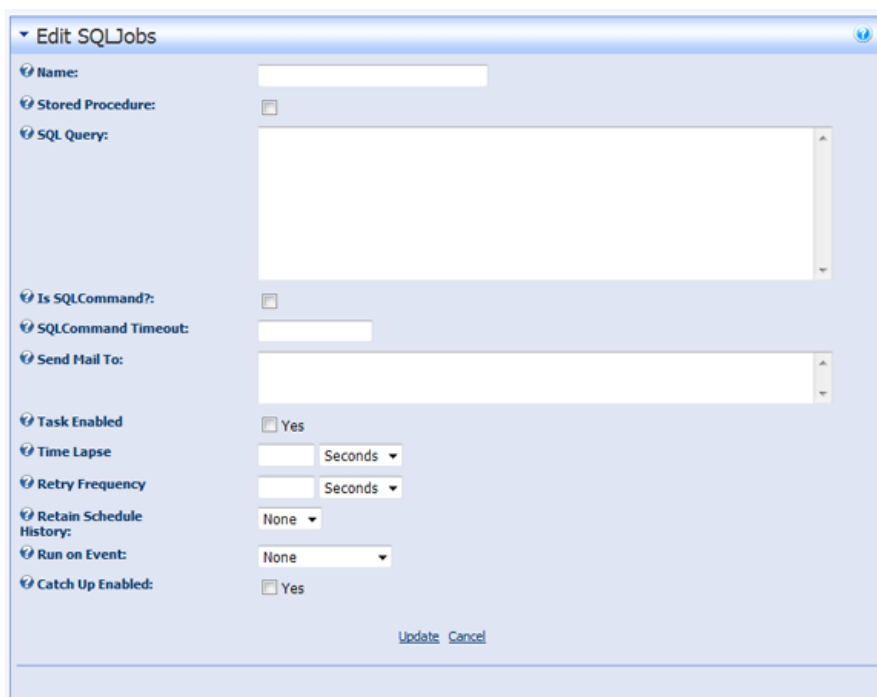
Una vez que haya instalado el módulo y lo agregue a una página verá algo como:



En este momento todavía no ha agregado ninguna tarea, el módulo necesita configurarse. Pulse el enlace para agregar una tarea

GESTIÓN DE TAREAS

Las tareas se configuran desde la pantalla de edición o adición



Nombre: Un nombre único para esta tarea

Procedimiento Almacenado: Si marca esta caja, aparecerá un desplegable para que seleccione un procedimiento almacenado existente en la base de datos.

SQL Query: Si esta caja está activa ud. puede escribir aquí la consulta a ejecutar.

Es SqlCommand: Si chequea esta caja, la consulta o procedimiento almacenado se ejecutará en modo SqlCommand siendo posible asignar un valor de Timeout para la ejecución del comando.

Exportar: Activando esta casilla aparece una caja de texto donde ha de introducirse la ruta física absoluta al fichero de exportación incluyendo la extensión. Para su comodidad se incluye una caja con la ruta física absoluta a la instalación de DotNetNuke, puede copiar y pegar este valor en la caja superior. Le recomendamos que exporte los ficheros dentro de la carpeta App_Data de su instalación, de esta forma el fichero no será accesible por cualquier usuario mediante el navegador.

Enviar Mail a: Escriba aquí una o varias direcciones separadas por “;” de esa forma, cada dirección recibirá un resumen tras la ejecución de cada tarea. Si la casilla exportar está activa, se incluirá el fichero de exportación adjunto al mensaje.

Tarea Activada: Marque esta caja para activar la ejecución periódica de la tarea, si no activa la caja el programador no ejecuta esta tarea.

Lapso de Tiempo: Configure cada cuanto tiempo quiere ejecutar la tarea

Frecuencia de reintento: Si la tarea falla cuando se ejecuta con la frecuencia establecida, se intentará ejecutarla de nuevo con este valor hasta que se ejecute correctamente.

Retener Histórico del Programador: Cada vez que se ejecuta la tarea, queda registrado en un histórico el resultado de la tarea, el tiempo de inicio, de final y duración de la ejecución. Esta entrada se refiere al número de entradas a almacenar en este registro histórico.

Ejecutar en Evento: Posibilita que la tarea se ejecute en otro evento distinto al del programador. Solamente está garantizada la ejecución en el evento Application_Start ya que Application_End no siempre está garantizado que vaya a ejecutarse en ASP.NET y DotNetNuke.

Recuperación habilitada: Si no se ha podido ejecutar la tarea por algún motivo, este evento posibilita que una vez recuperado el programador, la tarea se ejecute todas las veces que no lo haya hecho antes.

AGREGUE LA PRIMERA TAREA

Ejemplo de tarea

Como sabrá, las bases de datos SQL Server requieren realizar una compactación de forma periódica para evitar que el fichero de Log vaya creciendo sin límites. La instrucción SQL que realiza esta compactación es DBCCShrink(aquielnombredesubd)

Si se programa una tarea que ejecute este comando cada 30 días, nos estaremos asegurando de que nuestra base de datos esté siempre compactada y ofreciendo el máximo rendimiento.

Para agregar esta tarea pulse el botón “Agregar nueva tarea y rellene el formulario de tarea con los siguientes valores)

Nombre: Compactación periódica Base de datos

Procedimiento Almacenado: No lo marque

SQL Query: DBCCShrink(aquielnombredesubd)

Es SQLCommand: No marque.

Enviar Mail a: Si desea que se le notifique, escriba su dirección de correo

Exportar: No lo marque

Tarea Activada: Marque esta caja para activar la ejecución periódica de la tarea, si no activa la caja el programador no ejecuta esta tarea.

Lapso de Tiempo: Configure cada cuanto tiempo quiere ejecutar la tarea por ej. 15 o 30 días

Frecuencia de reintento: Si la tarea falla cuando se ejecuta con la frecuencia establecida, se intentará ejecutarla de nuevo con este valor hasta que se ejecute correctamente. Seleccione 2 horas

Retener Histórico del Programador: Cada vez que se ejecuta la tarea, queda registrado en un histórico el resultado de la tarea, el tiempo de inicio, de final y duración de la ejecución. Esta entrada se refiere al número de entradas a almacenar en este registro histórico, guarde 25 registros

Ejecutar en Evento: nada

Recuperación habilitada: nada

CÓMO EJECUTAR PROCEDIMIENTOS ALMACENADOS PASANDO PARAMETROS

Puede ejecutar procedimientos almacenados y pasar parámetros. Seleccione Consulta SQL y use algo similar a los siguiente (Cambie la variable portalid por su portalid y portalalias por su alias de portal)

```

DECLARE @PortalAlias nvarchar(250)
DECLARE @PortalID int
DECLARE @StartDate datetime
DECLARE @EndDate datetime

select @PortalAlias='www.susitioweb.com'
select @portalid = 2
select @StartDate = CONVERT(VARCHAR(10),GETDATE(),111)
select @EndDate = getdate() + 1

exec dnn_getsitelog2 @Portalid,@PortalAlias,@StartDate,@EndDate

```

Puede programar esta tarea de forma que se exporten y envíen los resultados por correo electrónico del visor de detalle del log con la periodicidad que desee.

PERSONALIZACIÓN AVANZADA DE TAREAS EN LA VERSIÓN FUENTE

Si dispone de la versión fuente del módulo dispone de la posibilidad de ejecutar una clase externa en Visual Basic con la que realizar múltiples operaciones adicionales en la ejecución de la tarea. Veamos cómo funciona.

Ud. instala la versión fuente de SQLJobs y en la pantalla de agregar o modificar tareas verá una opción llamada “Pasar a clase externa” active esta casilla.

La segunda parte es la programación avanzada de la tarea, busque en su carpeta [APP_Code] el fichero ExternalWork.vb y ábralo.

Repita la parte Case “name” cambiando “name” por el nombre de su tarea.

La parte Case “name” que se incluye es de ejemplo, para que vea como hacer algo dentro del Select principal, ud, debe agregar su propio código como otro “case” dentro del Select.