*AUTOMATIZACIÓN DE PROCESOS*

*JOB Manager*

*Documento de diseño*

Argentina

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Autor** | **Fecha** | **Detalle** |
| 1.0 | Leonardo Lagostena | 18/01/2017 | Primera versión |
|  |  |  |  |

## Especificación de Diseño Del Servicio Interno JOB Manager

Objetivo

Ante la necesidad de contar con un mecanismo de secuenciación y orquestación de servicios básica, se propone el desarrollo de un componente de servicio interno al SBM, que permita la articulación de diversos servicios comunes a los efectos de realizar tareas conjuntas de manera ordenada. A tal efecto, al conjunto de servicios estructurados en una secuencia de pasos específicos se lo denominará JOB. El sistema deberá proveer la capacidad de definir y registrar distintos JOBs y su composición, observando las restricciones y alcances del modelo de base de datos del SBM.

Prerrequisitos

Al ser un servicio interno, el mismo debe implementar la interfaz específica que el SBM requiere para estos casos, sobre todo en lo referido al consumo del componente de acceso a datos propio. Así mismo, es condición necesaria la implementación de un mecanismo de cancelación y desactivación del componente a demanda, sin que dependa de su eliminación activa por parte del SBM. La estabilidad, robustez y tolerancia a fallos es requisito clave para su implementación.

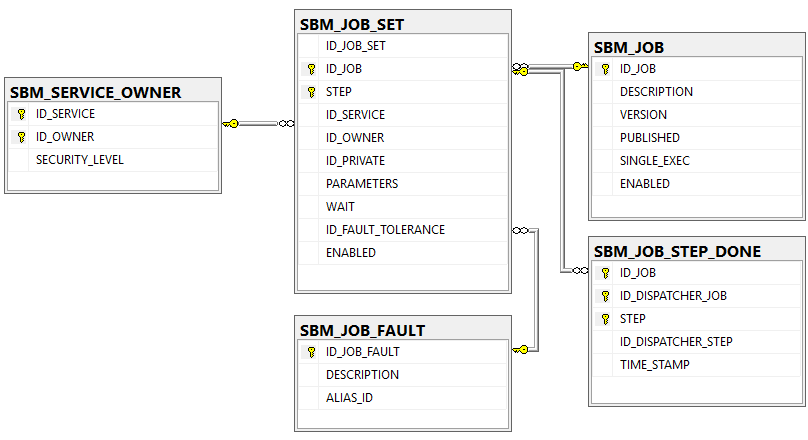
Descripción de la funcionalidad

Por tratarse de un servicio interno, su ejecución dependerá del mecanismo previsto a tal efecto dentro de la lógica interna del SBM. Considerando además que el componente debe tener acceso a la DB propia del sistema, como así también a valores y constantes de configuración o en general al estado interno del SBM. Adicionalmente, dentro de los parámetros de invocación originales informados al SBM, es mandatorio incluir el ID del JOB que se requiere ejecutar. Así mismo, y dada la posibilidad de que múltiples JOBs deban ser ejecutados simultáneamente, este servicio debe permitir paralelización en el momento de su instanciación.

Como se indicó, un JOB será un conjunto de uno o más servicios agrupados bajo un mismo identificador y denominación o nombre. Ese conjunto de servicios estará además ordenado en una secuencia o pasos, que determinará cómo los mismos deben ser invocados.

Para dar soporte a esta operatoria, será necesario contar con un conjunto de tablas internas que permitan conformar la composición de cada JOB, así como también dar seguimiento a todo el ciclo de ejecución de cada servicio en él incluido.

A continuación se muestra el DER de la solución:

Dónde:

* SBM\_JOB: Tabla maestra de JOBs
* SBM\_JOB\_SET: Composición de servicios en secuencia de cada JOB
* SBM\_JOB\_STEP\_DONE: Registro de invocaciones de servicios realizadas por JOB
* SBM\_SERVICE\_OWNER: Relación Service – Owner del SBM

La tabla SBM\_JOB registra todos los JOBs definidos en el sistema, indicando si los mismos son paralelizables y si están habilitados. En cuanto al atributo de paralelización (SINGLE\_EXEC) es fundamental destacar que si bien puede controlarse si más de un mismo JOB puede ejecutarse al mismo tiempo, son los servicios que lo componen los que en última instancia determinan si esto es o no posible. Por más que se especifique en un JOB que este puede ser paralelizado, si un solo servicio de su composición no soporta esta modalidad, es muy posible que se produzca la cancelación de todo el JOB, al fallar al intentar ejecutar más de una vez ese mismo servicio.

Seguidamente, la tabla SBM\_JOB\_SET guarda la estructuración de cada JOB registrado en la tabla maestra, como un conjunto de servicios ordenados según una secuencia de pasos preestablecidos, mediante el uso de un número de orden en el campo STEP. Cada servicio debe tener registrado en esa tabla todos los campos usuales para su correcta invocación, a saber: ID\_SERVICE, ID\_OWNER, ID\_PRIVATE (opcional) y PARAMETERS (opcional). Para configurar el tratamiento que el componente tendrá con cada uno de ellos, debe indicarse además el grado de tolerancia a fallas del mismo, y si después de ejecutar debe esperar a que finalice todo lo invocado hasta el momento (WAIT). Por supuesto el campo ENABLED, permite excluir un paso o secuencia de una manera simple, de ser necesario por algún motivo. Para cada ID\_SERVICE e ID\_OWNER se constatará la integridad referencial contra la tabla SBM\_SERVICE\_OWNER.

La tabla SBM\_JOB\_STEP\_DONE actuará en la práctica como un puntero de ejecución del JOB en curso, registrando en ella todas las invocaciones que se vayan realizando. Para cada registro encolado en la tabla SBM\_DISPATCHER, se guarda el correspondiente ID\_DISPATCHER obtenido, en el campo ID\_DISPATCHER\_STEP, completando además con el número de STEP originalmente asignado en la tabla SBM\_JOB\_SET, el ID\_JOB y el ID\_DISPATCHER que recibió el pedido que disparó la ejecución del JOB en el campo ID\_DISPATCHER\_JOB. Cada registro de esta tabla a su vez, tendrá un registro automático de su fecha de creación en el campo TIME\_STAMP.

Proceso

Como ya se indicó, los servicios internos son identificados con ID\_SERVICE = 0 y con ID\_OWNER = 0 al invocarse. Dentro del campo PARAMETERS de la tabla SBM\_DISPATCHER se especifican de manera opcional los parámetros que cada servicio pueda requerir para ejecutar. Para el caso de los servicios internos, ese campo es obligatorio y su contenido se debe corresponder siempre con este formato:

<?xml version="1.0" encoding="UTF-8"?>

<parameters>

<ID></ID>

<parameter></parameter>

</parameters>

Identificado entonces al servicio interno como correspondiente al JOB Manager (campo ID), dentro del campo parameter se debe especificar a su vez el ID del JOB a ejecutar. Como este servicio no requiere parámetros adicionales, simplemente bastará con especificar dentro de dicho campo, el identificador requerido, por ej: <parameter>12</parameter>. Siguiendo el ejemplo, se debe buscar en la tabla SBM\_JOB el registro que coincida con un ID\_JOB = 12, y si el mismo está habilitado (ENABLED) se procede con su ejecución.

La tabla SBM\_JOB\_SET contiene la composición estructurada en pasos de invocación de cada uno de los servicios que conforman el JOB, pero además y fundamentalmente, actúa como un secuenciador. Mediante el ordenamiento de cada registro a través del campo STEP, la tabla describe la secuencia de invocación que de manera estricta el JOB Manager debe seguir.

Con este criterio, al momento de iniciar la ejecución de un JOB, se recuperan en orden según el campo STEP todos los pasos que conforman el SET del JOB para el ID\_JOB establecido y que estén ENABLED (ver NOTA). Luego se lanzan progresivamente TODOS los procesos de esa lista hasta que se alcanza el final de la misma, o un registro tiene un campo con la directiva WAIT en True. En este último caso, ese mismo registro también es ejecutado, pero no se continúa con otra invocación hasta que TODOS los servicios pendientes de finalización concluyan (sin importar el orden en que finalicen). Entonces, mientras no se alcance el final del SET, se repite el proceso tantas veces como sea necesario, hasta que se alcanza efectivamente el último STEP especificado.

En este punto es fundamental comprender que la directiva WAIT indicada en cualquier STEP de la tabla SBM\_JOB\_SET, afecta tanto al mismo registro como a TODOS los precedentes, o dicho de otra manera, a todos los servicios del JOB en ejecución aún sin finalizar. Como concepto central también es importante notar que, independientemente de la existencia o no de la directiva WAIT en la última secuencia o paso de un SET, el sistema debe esperar siempre a concluir con TODOS los servicios lanzados, antes de dar por finalizada la ejecución del JOB actual. En la práctica sería como asumir que el último registro tiene invariablemente ese campo en true.

De esta manera, el campo ENDED de la tabla SBM\_DONE con el registro que originalmente invocó al JOB, coincidirá con la fecha de finalización de la última tarea que haya concluido, además del estado de terminación, que se reflejará según el criterio indicado para evaluar cada invocación (campo FAULT\_TOLERANCE) que el JOB realice. Esto permite reflejar correctamente el tiempo real empleado por todo el JOB para finalizar, evitando así realizar el cálculo sumando los tiempos parciales de cada servicio ejecutado.

Durante la ejecución de un JOB y a medida que los servicios van finalizando, es necesario ir verificando su estado con el campo ID\_DONE\_STATUS de la tabla SBM\_DONE, comparándolo con la configuración de la Tolerancia a Falla indicada en el registro correspondiente del SET. En el preciso instante en que una evaluación de esta característica determina que el JOB fracasa, es necesario CANCELAR TODOS los servicios del JOB que aún estén activos. Para ello forzamos su cancelación inmediata en la tabla SBM\_OBJ\_POOL, actualizando el campo MAX\_TIME\_RUN con valor cero. Inmediatamente el mismo servicio debe finalizar notificando su conclusión con errores.

NOTA: al momento de recuperar la lista o SET de servicios que conforma el JOB, sólo se toman aquellos registros que estén habilitados para su uso (ENABLED), pero no se verifica que el servicio en sí lo esté para que puede ser efectivamente ejecutado. Esa responsabilidad no es parte de este componente, correspondiendo al SBM y en última instancia, al que conformó el JOB su control. Por lo tanto si un servicio, o varios de ellos, no están habilitados para ejecutar al momento de su invocación, cancelarán con el error usual, debiendo entonces determinar si se puede continuar o no con la ejecución del JOB, según lo indique lo registrado en el campo ID\_FOULT\_TOLERANCE de cada registro del SET.

Diagrama del Proceso

A continuación se muestra el diagrama del proceso descripto anteriormente, agregando más detalles en cada punto de función.



**Donde:**

**P1**: Recupero todos los registros de la Tabla SBM\_JOB\_SET para el ID\_JOB indicado en orden ascendente por el campo STEP y sólo los que estén ENABLED.

**P2**: Con la lista de servicios a invocar, procedo a insertar en la tabla SBM\_DISPATCHER los servicios en el orden establecido tomando los siguientes campos: ID\_SERVICE, ID\_OWNER, ID\_PRIVATE y PARAMETERS. En este último campo debemos tener presente que si se trata de una referencia, debe procederse según lo indicado en (). Paralelamente inserto en la tabla SBM\_JOB\_STEP\_DONE el ID\_DISPATCHER de cada invocación en el campo ID\_DISPATCHER\_STEP junto con el STEP, el ID\_JOB y el ID\_DISPATCHER\_JOB que se debe corresponder con el ID\_DISPATCHER que el servicio original obtuvo. Continúo con este procedimiento hasta llegar al final de la lista (EOF) o encuentro un registro con la directiva WAIT en true.

**P3**: Usando los registros de la tabla SBM\_JOB\_STEP\_DONE para el ID\_DISPATCHER\_JOB actual, verifico por cada ID\_DISPATCHER\_STEP en la tabla SBM\_DONE. Si alguno terminó, tomo el estado con el campo ID\_DONE\_STATUS.

**P4**: Con el valor del campo ID\_DONE\_STATUS del servicio que finalizó, procedo a compararlo con el ID\_FOULT\_TOLERANCE indicado en el STEP, según el siguiente criterio: si es 1 el servicio debe concluir únicamente por éxito (Completed successfully), si es 2 puede ser alguna de las dos posibilidades: Completed successfully OR Completed with errors, finalmente si es 3 es indistinto ya que no se interesa corroborar su resultado.

**P5**: Usando los registros de la tabla SBM\_JOB\_STEP\_DONE para el ID\_DISPATCHER\_JOB actual, procedo a actualizar el campo MAX\_TIME\_RUN de la tabla SBM\_OBJ\_POOL con valor cero, según el campo ID\_DISPATCHER para cada uno de los ID\_DISPATCHER\_STEP que figuren activos.

**P6**: Actualizo en SBM\_DONE con ID\_DISPATCHER\_JOB campos ENDED con fecha actual, ID\_DONE\_STATUS con código apropiado (éxito o error según corresponda).

Cancel Request

Eventualmente, El SBM puede enviar una señal de cancelación inmediata a este servicio interno. Normalmente esa notificación sucede al establecer el valor cero en el campo MAX\_TIME\_RUN de la tabla SBM\_OBJ\_POOL. A diferencia de otros servicios, los internos en general y este en particular, serán notificados desde el SBM siendo exclusiva responsabilidad del mismo servicio proceder a detenerse sin demora. Como se indicó, antes de hacerlo debe proceder a cancelar todos los servicios activos del JOB en curso, concluyendo finalmente con su propia cancelación. El procedimiento para cancelar los servicios en ejecución será el usual de marcarlos para cancelación, poniendo el campo MAX\_TIME\_RUN en cero de la tabla SBM\_OBJ\_POOL. De esta manera, el SBM reconoce de manera automática que esos servicios deben ser detenidos.

Puesto que la responsabilidad de la cancelación del JOB en curso no depende del SBM, sino del mismo servicio, el SBM se limitará a dar por cumplida la solicitud luego de la correspondiente notificación, procediendo a establecer el estado de los registros involucrados como si se tratara de una cancelación normal.

Método para retroalimentación

El modelo actual del JOB manager implica que el conjunto de servicios y sus parámetros deben estar predefinidos y guardados de antemano en la tabla descriptora del set SBM\_JOB\_SET. Esto le imprime cierta rigidez a la solución, al no poder reutilizar el resultado de un servicio como entrada o parámetro de otro. Para minimizar esta limitante, se conviene en adoptar una convención al momento de especificar que un parámetro debe ser tomado desde el campo RESULT de la tabla SBM\_DONE.

Si en un servicio de la tabla SBM\_JOB\_SET se encuentra en el campo PARAMETERS la siguiente expresión: **{STEP=nn}**, quiere significar que se debe tomar el campo RESULT de la tabla SBM\_DONE del ID\_DISPATCHER que figure en el campo ID\_DISPATCHER\_STEP de la tabla SBM\_JOB\_STEP\_DONE para el STEP nn. El valor tomado será directo y tal cual se encuentre en el momento, sin realizar ningún trabajo de transformación o adaptación alguno. Lo cual implica que el campo RESULT tomado debe ser legible como parámetro del servicio a retroalimentar.