PBtoWS - Procesos: Guía Rápida de implementación del Componente Proxy Backend

Este capitulo contiene información relacionada con el proceso de creación de componentes proxy aplicando la arquitectura propuesta en el desarrollo backend. El objetivo de esta sección es centrar al desarrollador en los aspectos fundamentales que debe tener presente al crear componentes Proxies que serán utilizados en los consumos internos de servicios SOAP Powerbuilder. Tener presente que sólo se explicará el proceso de implementación y se excluirán los demás procesos asociados a la utilización. Para más información favor consultar los pasos del Check List Component

Paso 0: Tipos de Componente Proxy

Antes de iniciar el proceso de creación de la clase de consumo se debe identificar el tipo de proxy el cual puede ser de 2 Tipos:

- **ComponentProcessProxy** - Componente de Proceso Estandard: Este tipo de proxy se debe implementar para el consumo de servicios que representan procesos estandares del ERP.
- **ComponenteCrudProxy** - Componente de Proceso Crud: Este tipo de proxy se debe implementar para el consumo de servicios CRUD (simples).

Paso 1: Creación del Proyecto del Componente Proxy

El primer paso consiste en crear el proyecto del componente Proxy. La información relacionada de esta actividad puede ser consultada en el siguiente link Crear Componente Proxy, además el componente debe estar previamente configurado en el catalogo de configuración de componentes.

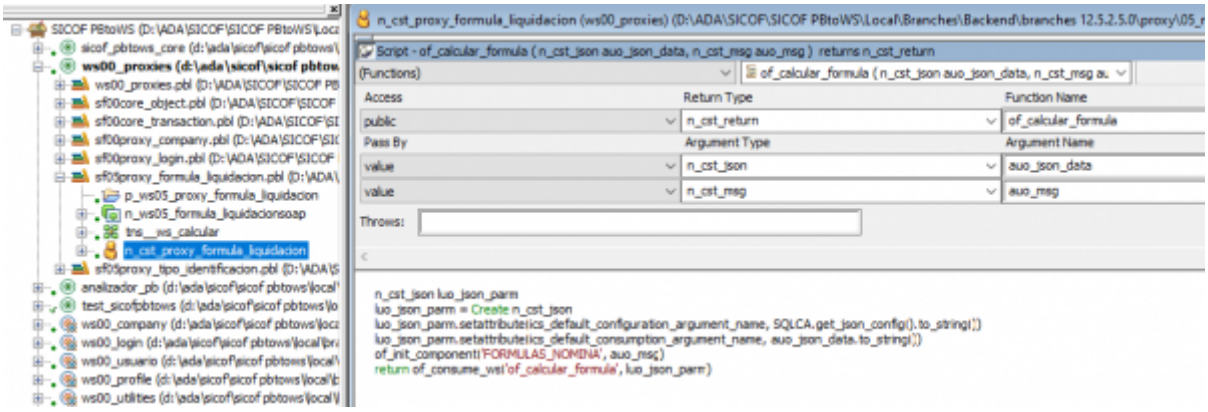
Paso 2: Crear la Clase del Componente Proxy

El siguiente paso consiste en definir la clase de operación del componente proxy de la sigueinte forma:

- Extender la clase **n_cst_proxy** para crear la clase de operación del componente proxy

Nota: Tener presente que la clase de operación debe agregarse a la libreria **sf[código de la aplicación]proxy_(nombre componente).pbl**.

A continuación se visualiza una imagen de ejemplo con la implementación de la clase de operación



1. **Clase de Operación**: Es la encarga de exponer el consumo del servicio web en funciones accesibles en el modelo de invocación de componentes. Se debe tener presente que estas clases en ningun escenario deben implementar lógica del negocio solo deben servir de intefaz de comunicación del consumo y respuesta de los servicios.

Paso 3: Modelo de implementación de invocación por capas

A continuación se explica con ejemplos en imagenes el modelo de implementación de invocación por las capas del framework aplicando la arquitectura propuesta. Se toman 2 casos en los cuales se pueden implementar Proxies.

Paso 3: Caso ComponentProcessProxy (Ejemplo Componente Fórmulas Liquidación Nómina)

Este caso expone la forma adecuada de creación de clases de operación de proxy para procesos del ERP.

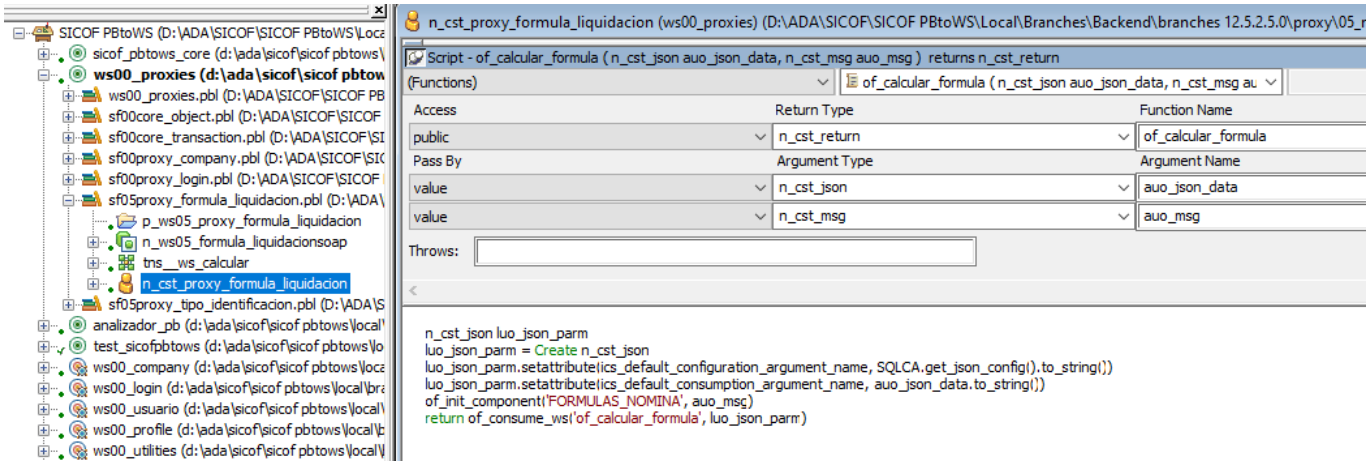
Modelo de implementación: Método Wrapper

El modelo debe implementar un método wrapper[1)] que permita:

1. Inicializar el Componente de consumo del Proxy (Previamente debe estar configurado en el catalogo de componentes. El método **of_init_component** recibe el nombre del componente y la referencia del objeto de mensajes.
2. Crear el json de parametros que recibe el servicio el cuál en la mayoría de los casos debe recibir la cadena json del consumop actual del consumo **as_config** y una cadena en formato json que especifica los datos que requiere el consumo.
3. Por último debe consumir la interface **of_consume_ws** donde siempre se debe enviar el método actual y el json con los parametros de consumo del servicio.

De igual forma debe asegurar que el resultado del proceso debe ser devuelto en un objeto tipo **n_cst_return**

A continuación se visualiza la imagen de la implementación del Componente Fórmula Liquidación.



Consideraciones

- No implementar lógica de negocio en las clases n_cst_proxy

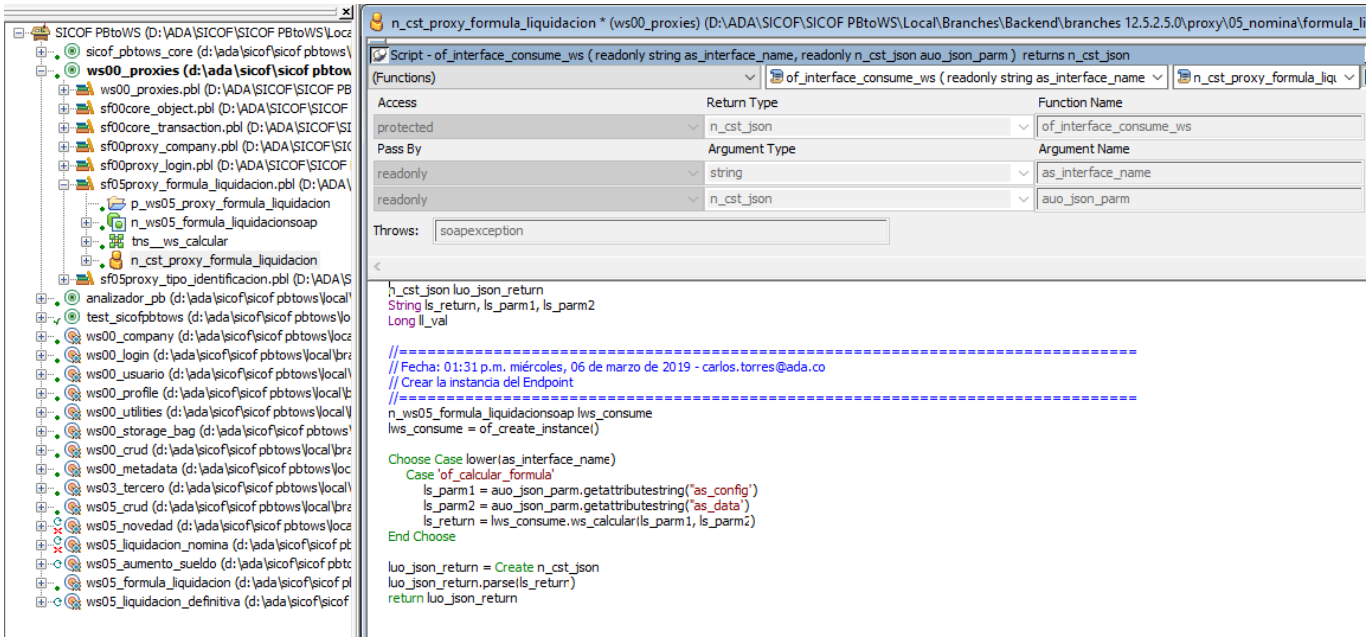
Modelo de implementación: Método of_interface_consume_ws

En este método se debe implementar la lógica de consumo de las operaciones expuestas por el servicio. Se deben seguir los siguientes pasos:

- Definir el objeto proxy del componente el cual contiene los métodos expuestos por el servicio por lo general el nombre es generado automaticamente y está definido de la siguiente forma **n_ws[código de la aplicación]_[nombre del componente]soap** Ejemplo: n_ws05_formula_liquidacionsoap
- Inicializar el objeto proxy consumiendo el método **of_create_instance()**
- Crear un Choose Case donde se puedan identifcar las interfaces de operación del servicio las cuales estan definidas en los métodos Wrapper creados. Cada opción del case debe consumir un método del proxy y debe devolver el resultado en una variable de tipo String
- Se debe encapsular el resultado y devolver en un tipo **n_cst_json** para que sea procesado en las capas de implementación posteriores.

De igual forma debe asegurar que el resultado del proceso debe ser devuelto en un objeto tipo **n_cst_json**

A continuación se visualiza la imagen de la implementación del Componente Fórmula Liquidación.



Consideraciones

- No implementar lógica de negocio en las clases n_cst_proxy

1)

http://consejostallerdeprogramacion.blogspot.com/2015/11/que-es-wrapper-y-donde-lo-utilizamos.html

From:
http://wiki.adacsc.co/ - **Wiki**

Permanent link:
http://wiki.adacsc.co/doku.php?id=ada:tips:sicoferp:general:pbtows:procesos:guiarapidacomponenteproxy&rev=1567459999

Last update: **2019/09/02 21:33**