

PBtoWS - Procesos: Guía Rápida de implementación de la Arquitectura Backend

Este capítulo contiene información relacionada con el proceso de creación de componentes 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 que serán expuestos en servicios SOAP Powerbuilder. Tener presente que sólo se explicara el proceso de implementación de la arquitectura y se excluirán los demás procesos asociados a la creación. Para más información favor consultar los pasos del [Check List Component](#)

Paso 1: Creación del Proyecto

El primer paso consiste en definir la estructura del repositorio del componente. La información relacionada oncesta actividad puede ser consultada en el siguiente link [Crear Componente](#)

Paso 2: Establecer la Clase Transacción Proyecto

Una vez creada la estructura del componente el siguiente paso consiste en redefinir la clase que mapeará las conexiones de la base de datos. Esto es necesario ya que la transacción desempeña un papel fundamental para el procesamiento de la información. La clase encargada de ese proceso es **n_cst_transaction** y está en la librería **sf00core_object.pbl**. Ubíquese en el objeto **application** del proyecto y presione el botón **Additional Properties** para desplegar la ventana de propiedades adicionales del proyecto luego seleccione la pestaña **Variable Types** y en el campo **SQLCA** cambie el valor **transaction** por **n_cst_transaction** aplique los cambios y presione el botón **Ok**. De esta forma ya quedará definida la clase transaction del componente.

Paso 3: Crear las Clases Base del Componente

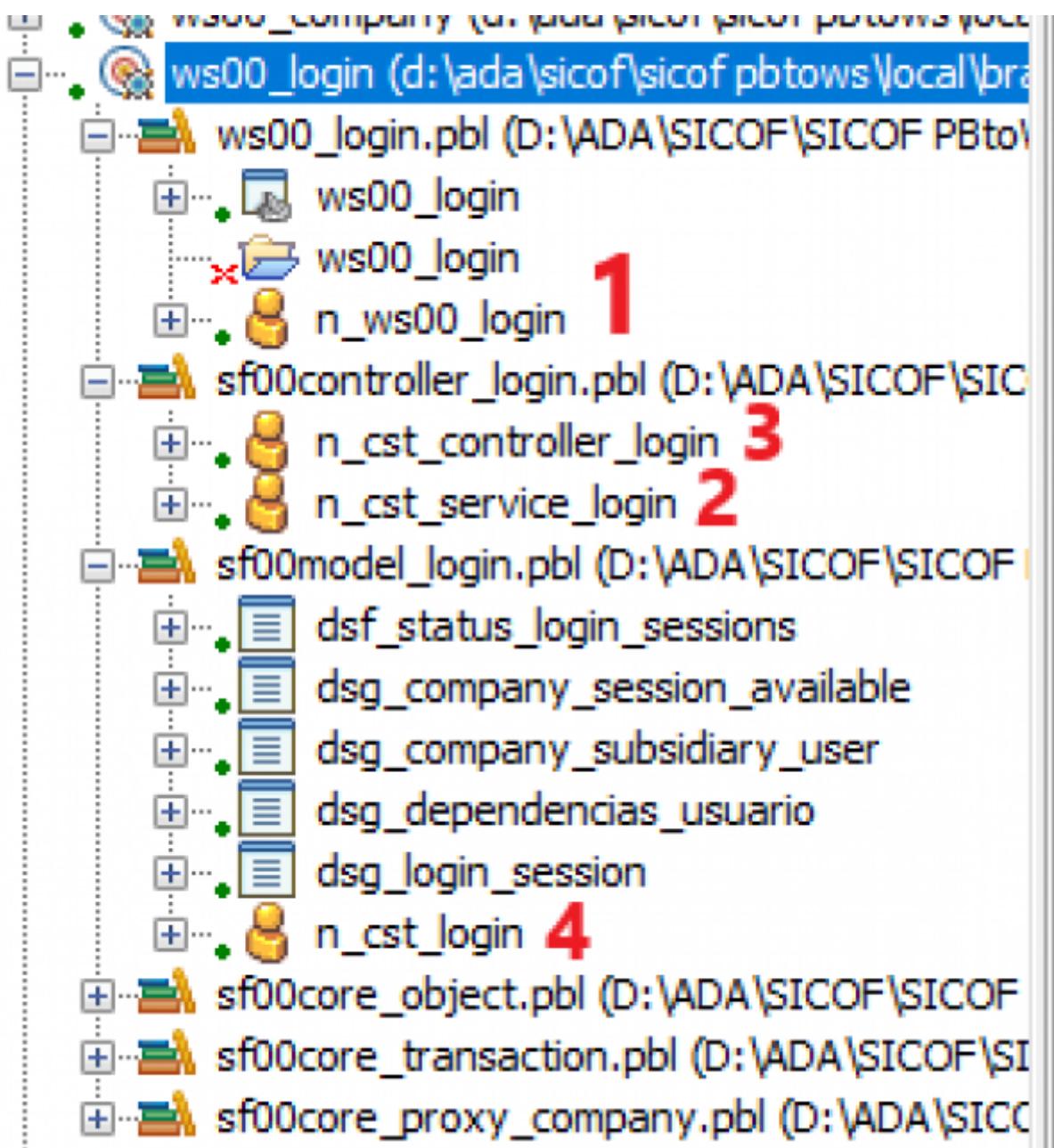
El siguiente paso consiste en definir las clases base de operación del componente las cuales deben ser extendidas (heredadas) del paquete de [Resolución y Orquestación de Servicios](#) de la siguiente forma:

- Extender la clase **n_cst_service** para crear la clase de invocación de servicios
- Extender la clase **n_cst_controller_process** para crear los controladores que serán utilizados por las clases invocadoras **n_cst_service**
- Extender la clase **n_cst_model** para implementar la lógica del negocio del componente, es decir en estas clases se generará el código fuente.

Nota: Tener presente que las clases invocadoras y controladoras deben agregarse a la librería sf(XX)controller_(xxxxxxxx).pbl y las clases del modelo ó lógica del negocio deben ser agregadas a la librería sf(XX)model_(xxxxxxxx).pbl.

A continuación se visualiza una imagen de ejemplo con la implementación de las clases base (tener

presente el numero asociado a cada clase para identificar el tipo)



1. **Clase Lanzadora:** Es generada automáticamente al crear el proyecto en ella se registran los métodos que serán expuestos en el servicio.
2. **Clase Invocadora:** Es la clase que va a ser invocada por la Clase Lanzadora, crea los controladores y lanza los métodos que inician los procesos.
3. **Clase Controladora:** Este tipo de clases realizan validaciones, provienen métodos de utilidades e invocan las clases de la lógica del negocio.
4. **Clase de Logica del Negocio:** Contiene el código powerbuilder de los procesos del ERP.

Paso 4: Modelo de implementación de invocación por capas

A continuación se explica con ejemplos en imágenes el modelo de implementación de invocación por las capas del framework aplicando la arquitectura propuesta. El componente de referencia es el login.

Modelo de implementación Clase Lanzadora

```

STCOP PBtoWS (D:\ADA\STCOP\STCOP PBtoWS\loc
  sicoft_pbtows_core (d:\ada\sicoft\sicoft_pbtows\)
  ws00_company (d:\ada\seco\pbtows\local\bni
    ws00_login.pbl (D:\ADA\STCOP\STCOP PBto
      ws00_login
        n_ws00_login
      sfc00controller_Login.pbl (D:\ADA\STCOP\STCOP
        n_cat_controller_login
        n_cat_service_login
      sf00model_login.pbl (D:\ADA\STCOP\STCOP
        dfl_status_login_sessions
        dsg_company_session_available
        dsg_company_subsidary_user
        dsg_dependencias_usuarios
        dsg_login_session
        n_cst_login
      sf00core_object.pbl (D:\ADA\STCOP\STCOP
      sf00core_transaction.pbl (D:\ADA\STCOP\STCOP
      sf00core_proxy_company.pbl (D:\ADA\STCOP\STCOP
        ws00_login (string as_config) returns string
        Copyright 2018 ADA, Inc. All rights reserved.
        Function/Event: ws_login
        Description: Proceso que realiza el inicio de sesión
        Arguments: as_config - Contiene los parametros del inicio de sesión login,password
        Returns: Cadena que indica el resultado del inicio de sesión.
        Author: ADA - carlos.torres@ada.co
        Date: 09:56 a.m., jueves, 24 de mayo de 2018
        Revision History: 1.0 - carlos.torres@ada.co - 24/05/2018 09:56:29 ; Initial version
        n_cat_factory luo_launch
        luo_launch = Create n_cst_factory
        return luo_launch.event factory_launch_event(n_cst_service_login', 'ue_login', as_config, luo_launch.ics_no_argument_required, luo_launch.ics_custom_access_validator)
  
```

From:

<http://wiki.adacsc.co/> - Wiki

Permanent link:

<http://wiki.adacsc.co/doku.php?id=ada:tips:sicoferp:general:pbtows:procesos:guiarapidacomponente&rev=1565630022>

Last update: 2019/08/12 17:13

