

Tutorial del servicio de exposición de funciones de Engage SrvEngage (V1)

Aspectos no cubiertos en por el tutorial:

- Visual studio, Framework de .Net, lenguajes de programación o IIS.
- WCF o protocolos involucrados en la comunicación.
- EngageIntegrationServices o utilitarios asociados.

Resumen del contenido del Tutorial:

- Qué es SvcEngage.
- Requisitos previos para desarrollar y/o poner productivo el servicio, ya sea software de base o conocimientos.
- Configuración básica de WCF para el servicio SrvEngage y para consumir EKS y EPS como cliente.
- Configuración básica del sitio web IIS, para instalar el servicio y para hacer que funcione con https.
- Estructura de servicios de Engage y los Apis respectivos.
- Estructura interna del web service:
 - o Estructura de la solución de Visual Studio.
 - o Librerías referenciadas.
 - o Clases de contexto global.
 - o Clases base e interfaces a implementar para agregar nuevos métodos web al servicio.
 - o Parámetros de configuración del servicio.
 - o Esquema del código de cada método web.
 - o Consumo de funciones de Engage a través de los Apis.
 - o Ejecución de Queries y Stored Procedures en forma directa.
 - o Clases y funciones de soporte.
 - o Mantenimiento automático de la sesión Engage.
 - o Tratamiento de errores y formato de la excepción estándar.
 - o Log de errores o de eventos.
- Ejemplo de generación de clases proxy con EISProxyManager.
- Consumo del servicio a través de EISProxyManager.

Qué es SvcEngage.

Es un servicio web de código abierto basado en Windows Communication Foundation (WCF). Se utiliza para exponer funciones del motor de Engage a través de métodos web, que serán invocados desde otras aplicaciones.

Requisitos previos para desarrollar y/o poner productivo el servicio, ya sea software de base o conocimientos.

- **Software para ejecución:** IIS 7.5 (o superior compatible previa certificación) y Framework 4.0 de .Net
- **Software para desarrollo:** Visual Studio .Net 2010 (o superior previa migración de los proyectos) y Framework 4.0 de .Net
- **Conocimientos de programación:** Manejo de Visual Studio, Visual Basic .Net, programación orientada a objetos
- **Conceptos básicos de web services:** WCF y protocolos involucrados en la comunicación y transmisión de datos (HTTP/HTTPS, XML, XSD, SOAP, WSDL, etc)
- **Manejo de utilitarios de Engage (opcional):** EISProxyManager

Configuración básica de WCF para el servicio SrvEngage y para consumir EKS y EPS como cliente.

La configuración se realiza en el archivo Web.Config, dentro del nodo `<system.servicemodel/>`. Tomando el archivo que sale de fábrica, los subnodos específicos para el servicio son `<services/>` y `<behaviors/>`, y para la definición de los clientes de EKS y EPS se utiliza en subnodo `<client/>`. Finalmente, en el subnodo `<bindings/>` se definen los enlaces tanto para el servicio como para los clientes. Tanto el contenido de los subnodos como los valores posibles de los parámetros están documentados por Microsoft.

Configuración básica del sitio web IIS, para instalar el servicio y para hacer que funcione con https.

Para instalar el servicio hay que definir una aplicación web de IIS y apuntarla a la carpeta raíz del sitio del servicio. La configuración de la aplicación web queda a consideración del administrador de IIS, según la carga de trabajo y el patrón de uso que tendrá. Para que el servicio funcione con https, hay que instalar un certificado de firma digital, crear el enlace https y asociarlo al certificado. Luego, se podrá optar por forzar el acceso únicamente via https o admitir también http.

Estructura de servicios de Engage y los Apis respectivos.

Los servicios de Engage con los que mayormente se va a interactuar son EngageKernelService (EKS) y EngagePassportService (EPS). El primero se encarga de las funciones de workflow y el segundo de la autenticación de usuarios y el mantenimiento de la sesión. Ambos servicios cuentan con un cliente API para poder ejecutar las funciones primitivas desde otra aplicación. Dichos clientes están

incluidos en las librerías EKSCientLibrary.dll y EPSCientLibrary.dll.

Estructura de la solución de Visual Studio.

La solución de Visual Studio contiene dos proyectos:

- **Sitio web del servicio:** Es la aplicación web principal, que permite exponer al servicio a través de IIS. Contiene las siguientes subcarpetas: **App_Code**, con código que puede sobrescribirse en la implementación; **App_Data**, con archivos de datos; **Bin**, con las dlls que contienen el código fijo que no debe alterarse en la implementación; **Log**, donde se guardarán los archivos de log.
- **ClassLibrary:** Librería de clases principal, que incluye todas las funciones que quedarán compiladas y no modificables dentro de una dll, y que se utilizarán desde el sitio web del servicio.

Librerías referenciadas.

Las dlls que se referencian desde el servicio son:

- **Clientes:** EKSCientLibrary.dll y EPSCientLibrary.dll
- **De parámetros globales:** EngageContext.dll
- **De manejo de persistencia:** EngagePersistence.dll
- **De funciones comunes:** EngageClassLibrary.dll
- **De terceros:** Oracle.DataAccess.dll

Clases de contexto global.

En el contexto global se guardan datos que serán accesibles desde todo el código del servicio en tiempo de ejecución. El contexto es un singleton que se instancia en la primera llamada al servicio y que dura hasta que se reinicia la aplicación web en el IIS. Las clases del contexto en orden jerárquico y de abstracción son:

- **clsAppBaseContext:** Clase abstracta con datos básicos de la aplicación y que implementa la interface `IDisposable`.
- **clsAppWsContext:** Clase abstracta que contiene parámetros recuperados inicialmente del archivo `Web.config`, las instancias de los clientes de EKS y EPS, el ticket de sesión y funciones primitivas para devolver esos datos privados. También contiene la función para obtener y validar el ticket de sesión.
- **clsAppContext:** Clase final que implementa el singleton y contiene la función para inicialización de parámetros de última instancia, ya sea por orden o por costo.

Clases base e interfaces a implementar para agregar nuevos métodos web al servicio.

Los objetos de programación que implementan la funcionalidad del servicio web son los siguientes:

- **Interface intSvcEngage:** Contiene el prototipo de todos los métodos web que expone el servicio.
- **Clase abstracta clsSvcEngageBase:** Implementa intSvcEngage utilizando métodos abstractos, contiene la instancia del contexto global, funciones primitivas para exponer el contenido del contexto y otras funciones comunes para ser utilizadas desde la clase final.
- **Clase final clsSvcEngage:** Hereda de clsSvcEngageBase e implementa todos los métodos abstractos.

Parámetros de configuración del servicio.

Los parámetros del servicio se definen dentro del nodo **<appSettings/>** del archivo Web.config y son los siguientes:

- **PersistenceCfgFile:** Path y nombre del archivo xml de configuración de la persistencia.
- **EPSEndpointName:** Nombre del Endpoint que usará el cliente EPS para comunicarse con el servicio de Passport.
- **EKSEndpointName:** Nombre del Endpoint que usará el cliente EKS para comunicarse con el servicio de Kernel.
- **EngageUser:** Usuario de Engage para que el servicio se registre en el servicio de Passport e inicie una sesión.
- **EngagePassword:** Clave del usuario de Engage que utilizará el servicio (en caso que la forma de acceso lo requiera).
- **EngageUnit:** Unidad que utilizará el usuario de Engage.

Esquema del código de cada método web.

El pseudocódigo del esquema básico que debería tener el código cada método web es el siguiente:

```
Try
    LogMessage("Inicio del método X.")
    Validación_De_Parámetros_De_Entrada
    Funcionalidad_Principal_Del_Método_X
    Retorno_De_Resultados_Si_Corresponde
Catch ex As Exception
    Throw LogException(New SvcEngageException(ex.Message))
Finally
    LogMessage("Fin del método X.")
End Try
```

Consumo de funciones de Engage a través de los Apis.

Para consumir funciones de Engage se utilizan las primitivas de los clientes EKS y EPS. Las instancias de esos clientes se mantienen automáticamente y se exponen a través de las funciones GetEksClientInstance y GetEpsClientInstance de la clase abstracta clsSvcEngageBase. Todos los métodos y parámetros de los clientes de EKS y EPS están disponibles a través del IntelliSense del editor de Visual Studio.

Ejecución de Queries y Stored Procedures en forma directa.

La abstracta `clsSvcEngageBase` contiene las siguientes funciones que se pueden utilizar para ejecutar queries y stored procedures:

- **ExecuteReader**: Ejecuta un query que devuelve registros.
- **ExecuteNonQuery**: Ejecuta un query que no devuelve registros.
- **ExecuteProcedureReader**: Ejecuta un SP que devuelve registros.
- **ExecuteProcedureNonQuery**: Ejecuta un SP que no devuelve registros.

El mantenimiento de la conexión a la base de datos es automático.

Clases y funciones de soporte.

Además de las funciones que se encuentran dentro del contexto Global o de las clases base del servicio, existen otras funciones comunes en la librería `EngageClassLibrary.dll` que pueden ser reutilizadas. Dentro de dicha librería también se encuentran clases de transporte que utilizan los clientes de EKS y EPS.

Mantenimiento automático de la sesión Engage.

En el servicio que sale de fábrica se incluye la lógica para la registración con un usuario de Engage específico y la obtención de un ID de sesión que perdura hasta que se reinicie la aplicación web en IIS. Este mecanismo hará que todas las llamadas a los métodos web utilicen el mismo ID de sesión cuando interactúan con EKS y EPS. En caso de ser necesario, se puede alterar el código para anular esta funcionalidad y obligar a la aplicación cliente del servicio que obtenga un ticket de sesión antes de ejecutar un método web. De esta manera, cada aplicación que consuma al servicio tendrá un ID de sesión distinto. Es decir, la que se registrará en Engage es la aplicación cliente y no el servicio. Si bien este nuevo mecanismo permite la identificación de cada aplicación, es más costoso a nivel performance y más complicado de implementar del lado de la aplicación cliente.

Tratamiento de errores y formato de la excepción estándar.

Para el transporte de los datos de excepciones desde el servicio hacia la aplicación cliente se utilizan clases especiales de excepciones. Por ello, es importante que cada método web devuelva ese tipo de excepciones y no otras. La clase principal de la excepción se llama `SvcEngageException`, y la clase que transporta el detalle de la excepción se llama `SvcEngageExceptionDetailType`. Ambas clases se encuentran dentro del archivo `clsExcExceptions.vb` del proyecto de la librería `ClassLibrary`.

Log de errores o de eventos.

El archivo de log del servicio tiene las mismas características que el del resto de los servicios de Engage, incluso en la forma en que se configura.

Ejemplo de generación de clases proxy con EISProxyManager.

Ver en forma práctica.

Consumo del servicio a través de EISProxyManager.

Ver en forma práctica.