Business Services - SVC

Representan cualquier tipo de servicio, administrados por el Dispatcher ya sean transaccionales o de consulta, síncronos o asíncronos.

En el framework fwk los servicios se envían desde las aplicaciones a través del wrapper o service connector bajo el siguiente patrón (*se establecen un patrón para ejecución de servicios*)

Patrón Ejecución síncrona:

```
[ServiceName]Res = Execute([ServiceName]Req)
```

Patrón Ejecución asíncrona:

```
ExecuteAsync(CallBack,[ServiceName]Req)
CallBack(result)
{
        [ServiceName]Response = result
}
```

Antes de pasar a definir los diferentes tipos de servicios es importante comenzar a explicar técnicamente los componentes que lo conforman.

Request y Response

Los Request y los Response son clases bases del framework que representan la interfaz de un servicio BusinessService. Están especialmente diseñadas para que sean procesadas por el despachador de servicios.

Tanto los Request como los Response son clases genéricas cuya interfaz de declaración es generic:

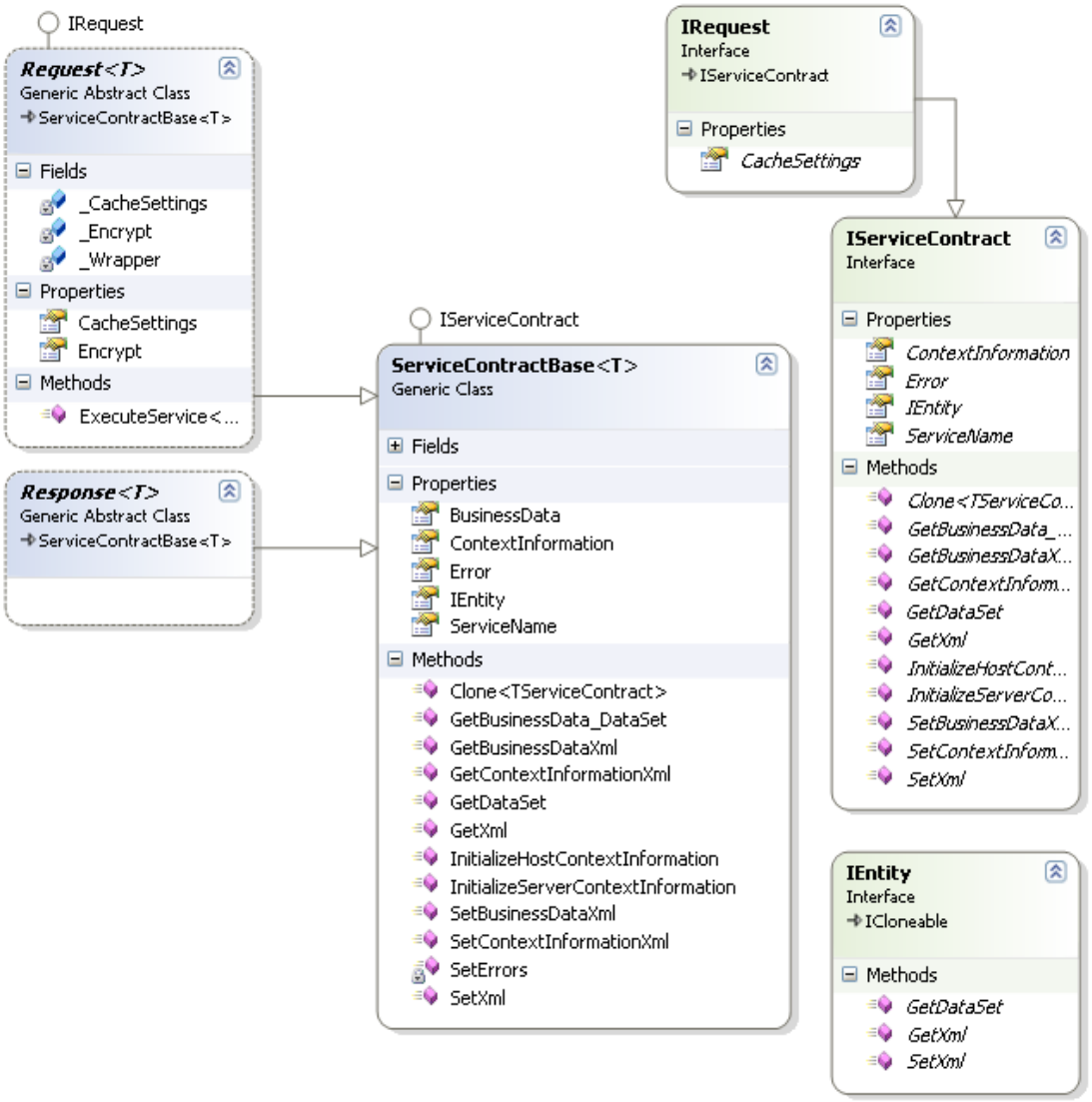
[ServiceName]Req : Request<T>
[ServiceName]Res : Response<T>

Done la clase **T** que recibe de forma genérica es cualquier entidad o clase que implemente de `IEntity.`

Esta clase, que implementa `IEntity,` es la que realmente tiene toda la estructura de negocio que necesita el servicio.

Nota: *cuando se definan servicios de alta complejidad la estructuras de las clases* `Entity/Entities` *que reciben los* `Request` *o* `Responses` *deberían ser representados por esquemas XSD. Esto es para disponer de una vista mas rápida y cómoda de estos objetos y además para que puedan ser utilizados para generar clases automáticamente.*

Diagrama de clase los contratos de servicios
A modo informativo mostramos el diseño de la interfaz deservicio en el framework



Los `Request` y `Response` que los desarrolladores generen serán clases que no tienen comportamiento ni atributos alguno en la definición de su cuerpo ya que las funcionalidades mas habituales están absorbidas por los métodos heredados de sus clases bases.

```
public class CrearFacturaReq : Request<FacturaBE>
{
}
```

Definir una clase que herede de `Request` y `Response` es para respetar el patrón de ejecución de un servicio contra un Dispatcher. No es solo una convención estándar, el despachador de servicio utilizara la herencia de estas clases para sus propósitos y asegurar la ejecución y correcta respuesta de la petición.

Alguno de los métodos o atributos heredados son:

Método o Atributo	Descripción
BusinessData	Este atributo representa la clase contenida dentro del Request o Response que hereda de Entity y contiene toda información funcional o de negocio.
~~SetXml(string pXMLService)~~	~~Rellena el Request o Response con la información del xml. Contiene tanto información de contexto como la funcional.~~
GetXml()	Obtiene el xml del Request o response.. Contiene tanto información de contexto como la funcional
ContextInformation	Información de contexto acerca del Request o response del servicio.
SetContextInformationXml()	Inicializa los datos contexto que pertenecen al Request con el contenido del xml.-
GetContextInformationXml()	Retorna el xml del Información de contexto que pertenece al Request o Response.-
InitializeServerContextInformation()	Establece la información de contexto del Request o Response del lado del despachador de servicios.-
InitializeHostContextInformation()	Establece la información de contexto del Request o Response del lado del cliente.-
SetBusinessDataXml(string pXMLData)	Inicializa los datos de negocio que pertenecen al Param o Result con el contenido del xml, dependiendo de si se trata de un Request o Response respectivamente.
GetBusinessDataXml()	Retorna el xml del Param o Result que pertenece al Request o Response respectivamente.-
Errors	Solo valido para los objetos que heredan de Response. Y representa cualquier tipo de error ocurrido desde que el despachador de servicio lanzo la ejecución de algún BusinessService
`Encrypt`	Solo para los Request. Permite que la información viaje encriptada y retorne encriptada entre el cliente y servidor
`CacheSettings`	Información de cacheo de los servicios. No es necesario q el programador diseño su propio código de isolation. El framework lo hace automáticamente bajo parámetros estándares y configuraciones altamente flexibles.

Context Information

El atributo ContextInformation es muy importante para obtener alguna información extra acerca del estado del servicio.

La siguiente tabla lista los atributos que dispone:

Propiedad	Descripción
HostName	Indica el host que inicio la peticion del servicio .
ServerName	Indica el server que atendio el servicio servicio .
UserName	Indica el usuario logueado en el host que inicio el servicio.
ServerTime	Indica fecha y hora de inicio o fin del servicio del lado del Servidor o despachador de servicio. Para un Request : fecha y hora de inicio del lado del server . Para un Response: fecha y hora de finalizacion del lado del server .
HostTime	Indica fecha y hora de inicio o fin del servicio del lado del Cliente. Para un Request: fecha y hora de inicio del lado del cliente (horario de inicio desde el host). Para un Response: fecha y hora de finalizacion del lado del cliente (horario de llegada al host).