Fwk.Logging

BLOQUE DE LOGING

El bloque de loging que provee el Framework permite a las aplicaciones (Back-End o Front-End) disponer un conjunto de mecanismos que facilitan la escritura y recuperación de sucesos que ocurran.

Modelo de componentes

La ubicación de los componentes de logueo es en las Fwk.Logging y sus componentes más importantes se muestran en el grafico siguiente:

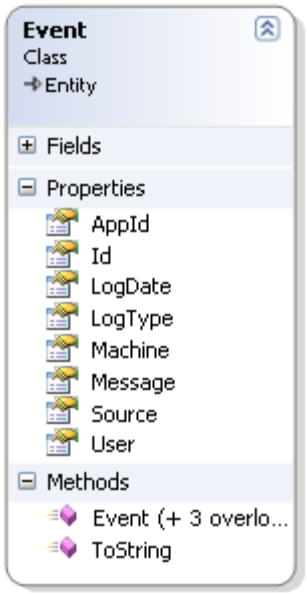

Figura 1.0

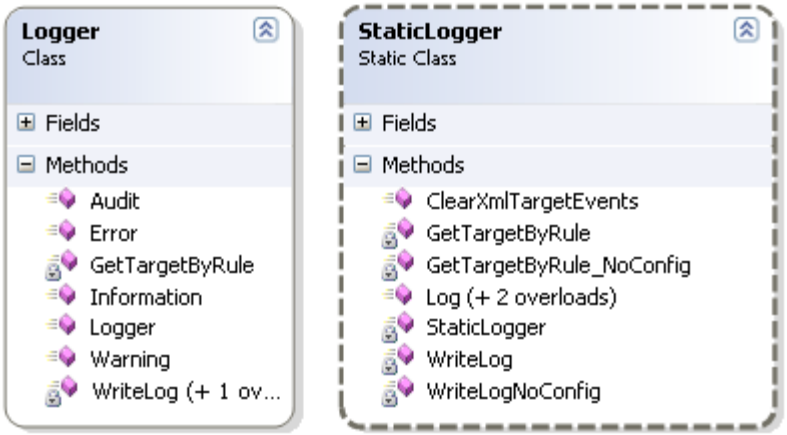

Figura 1.1

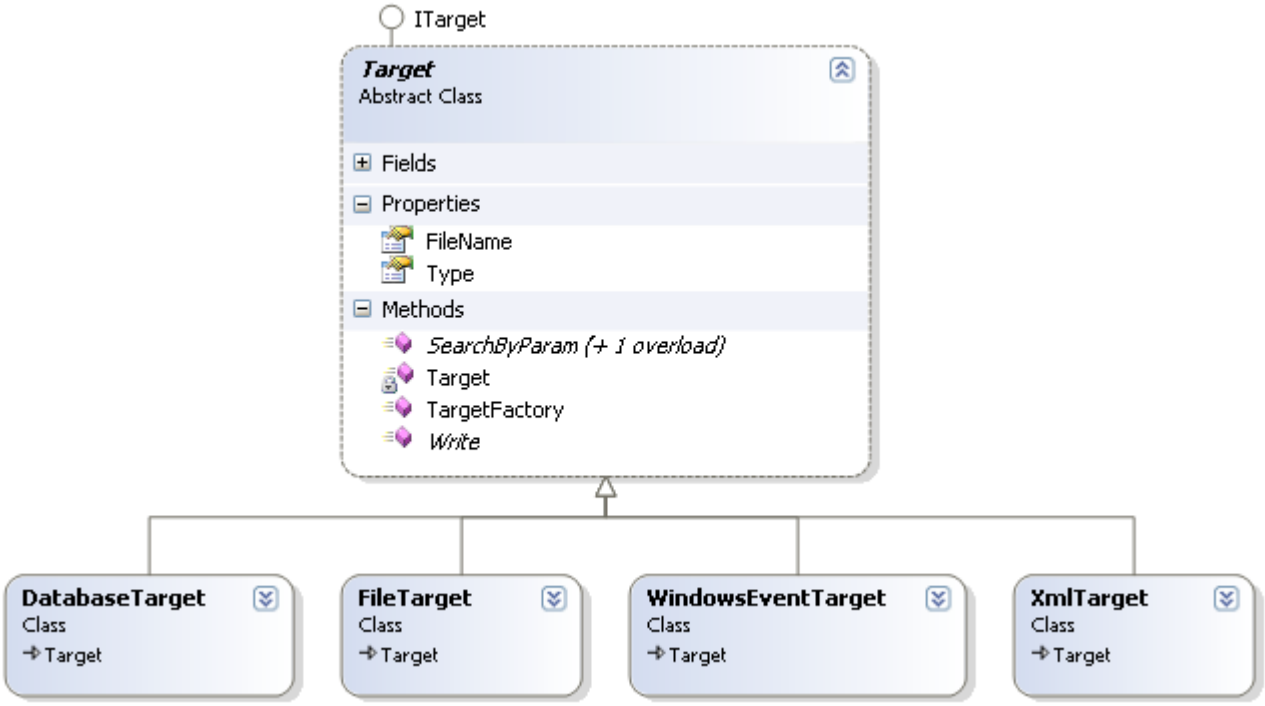

Figura 1.2

Descripción detallada de los componentes

Event (fig 1.0)

Representa el evento a loguear. El evento contiene la información relevante para que se desee persistir en las aplicaciones o servicios.-

Atributos

Nombre de atributo	Descripción
AppId	Identificador de la aplicacion o sistema qhe genera el log.-
Id	Identificador único del evento. Es un GUID
LogDate	Fecha y hora en la que se produce el evento.
LogType	Tipo de log o evento: 1. **Information**: Representa mensajes de información. 2. **Warning**, Representa mensajes de advertencia. 3. **Error**, Representa mensajes de error. 4. **None**, Representa la ausencia de tipo de evento. 5. **Audit**: Representa la ausencia de tipo de evento.
Machine	Equipo donde se origina el evento.
Message	Mensaje descriptivo del evento. Puede ser un xml que repreciente un objeto,. Es un tipo de dato `Fwk.Xml.CData` que protege los valores contenidos en el xml.
Source	Origen del evento.
User	Usuario que origina el evento.

Tabla 1.0

Targets

Estos componentes definen donde será almacenado el log o evento.

Los Tipos de targets. Son definidos por la enumeración TargetType

TargetType

Nombre del tipo	Fuente

File	Archivo plano
Xml	Archivo Xml
Database	Base de datos de SQL Server
WindowsEvent	Visor de suceso

Mecanismos de Logueo

Existen dos mecanismos para utilizar logueo. Por configuración o sin configuración.-

Ambas metodologías tienen sus ventajas y desventajas y su uso deberá ser evaluado por el equipo de desarrollo teniendo en cuenta los requerimientos y arquitectura del proyecto en cuestión.-

Por configuración

La metodología por configuración requiere que se agregue una sesión del framework en el archivo de configuración.-

Ventajas: Con esta metodología es posible hacer dinámico el logueo de las aplicaciones donde se podrá cambiar el Target de cada tipo de log sin necesidad de recompilar código

Desventajas: Requiere configuración previa en el archivo de configuración.-

Como se configura:

1) Agregue la siguiente sección en su archivo .config o web.config

```
section name="FwkLogging" type="Fwk.ConfigSection.LoggingSection, Fwk.Bases"/>
```

2) Cree las reglas de logueo que necesite agregando un nodo para dicha sección recientemente agregada:

```
<FwkLogging>
 <Rules>

 <add name="log1" events="Error"   target="Database"  cnnStringName="LogsDB"/>
 <add name="log2" events="Warning" target="Xml"  fileName="h:\logs\Logs.xml" />
 <add name="log3" events="Information"  target="WindowsEvent" />

 </Rules>
</FwkLogging>
```

De esta manera tenemos 3 reglas apara los diferentes tipos de logueo

- Log*1 dice: Todos los Errores almacenarlos en la base de datos, cuya cadena de conexión es LogsDB*
- log 2: Almacena todos los Warnings en un Xml *h:\logs\Logs.xml*
- *log 3: Dice que todo tipo de evento informativo deberá ser enviado el visor de eventos de Windows.-*

Nota: para mejor detalle de cómo configurar las reglas de logueo diríjase al Apéndice A: Configuración de reglas de logueo
Nota: para crear logs por medio de esta modalidad vea la sección Como se crea un log en este mismo documento

Sin configuración

Es posible utilizar directamente los componentes de logueo sin necesidad de configurar nada.

Esto es posible utilizando las sobrecargas del componente **StaticLogguer** que se detalla en la sección debajo

Como se crea un log

Las **ventajas** de este mecanismo son que permiten generar logs de diversas maneras sin necesidad de configuración alguna. Además permiten que se almacenen logs de un mismo tipo en diferentes orígenes de dato o Targets.

La **desventaja** es que al especificar por medio de código el tipo de evento y target esto queda, en una manera, harcodeado en los ensamblados de entrega y no se puede después cambiar sus valores sin compilar.-

El logueo sin configuración suele ser muy utilizado cuando cuando necesitamos generar eventos tipo informativos en servicios de Windows u otro tipo de servicios y que de antemano se sabe que jamás se modificara la forma en que almacena los logs.- Ej SQL Server anuncia errores de seguridad en Windows event

Como se crea un log

Para crear un log podemos usar el componente Logguer que requiere instanciacion o el componente StaticLoguer que levanta la configuracion po runica vez.-

Para el ejemplo utilizaremos StaticLogguer y su interfaz se visualiza en el grafico anterior Figura 1.1

StaticLogguer cuenta con un único método que es el que el desarrollador debe utilizar. Este método llamado Log se encarga de crear entradas en los diferentes targets.-

El mismo componente **StaticLogguer permite crear logs tanto de la forma preconfigurada y la sin configuración**

Esto lo hace atreves de sus sobrecargas en el método estático Log. En la siguiente tabla se dividen los las sobrecargas según requiera configuración o no.-

Sobrecarga	Requiere configuración	NO Requiere Configurable
`Log(Event eventType)`	X	

`Log(Event ev,`
`string path,`
`string fileNamePrefix)`
	X	

`Log(TargetType targetType,`
`Event eventType,`
`string fullFileName,`
`string cnnStringName)`
		X

Ejemplo utilizando configuración predeterminada:

Dada la configuración de reglas anterior, este codesnippet almacena un evento de error en la base de datos. El método Log en este caso recibe una clase Event que se detallo anteriormente (ver Tabla 1.0)

```
Event ev = new Event();
ev.Message.Text =  "Prueba de lgs";
ev.LogType = EventType.Error;
Fwk.Logging.StaticLogger.Log(ev);
```

Ejemplo utilizando Sin configuración predeterminada:

Este codesnippet almacena un evento de error en un archivo xml "c:\logs.xml"

```
Event ev = new Event();
ev.Message.Text =  "Prueba de lgs";
ev.LogType = EventType.Error;
StaticLogger.Log(TargetType.Xml, ev, @"c:\logs.xml",string.Empty);
```

Este codesnippet almacena un evento de error en la base de datos configurada en RRHHCnnString

```
Event ev = new Event();
ev.Message.Text =  "Prueba de lgs";
ev.LogType = EventType.Error;
StaticLogger.Log(TargetType.Xml,ev,string.Empty,"RRHHCnnString");
```

***Nota**: Para los tipos de logueo por archivo (File) A pesar de que la metodología Por configuración exija un nombre de archivo y esto defina un único archivo duro donde se almacenan los logs. Es posible violar esta regla anteponiendo un prefijo al nombre de archivo y también cambiar la ruta de destino. Para realizar esta variabilidad de logueo diríjase al Apéndice B: Crear log cambiando de prefijo en el nombre de archivo*

Apéndice A: Configuración de reglas de logueo

Para configurar las reglas de logue es necesario tener en cuenta el elemento de configuración que dispone de los siguientes atributos

Nombre del tipo	Fuente
name	Nombre de la regla. Puede ser cualquier nombre pero no se puede repetir
events	Corresponde a la enumeración **TargetType** • Information: Representa mensajes de información. • Warning, Representa mensajes de advertencia. • Error, Representa mensajes de error. • Audit: Representa la ausencia de tipo de evento. •
target	Donde se almacena el evento. Corresponde a la enumeración TargetType • File • Xml • Database • WindowsEvent
cnnStringName	Nombre de cadena de coineccion si el target es = Database
fileName	Nombre de archivo si el target es File o Xml

Configuración según tipo de almacén o Target

Database: Almacén en base de datos:

Los requisitos necesarios para almacenar los eventos en base de datos son:

1- Generar una regla que apunte que informe que dicho tipo de log se almacenara en base de datos

```
<add name="Rule1" events="Error"   target="Database"  cnnStringName="LogsDB"/>
```

2 – Contar con una cadena de conexión configurada en el archivo de configuración con el nombre que indica la Regla `cnnStringName="LogsDB"/>`

3- La base de datos debe contar con la tabla de almacen de logs: por lo tanto requerira que se corra el siguiente script en la base de datos:

%ProjectsFolder%\fwk\trunk\3.1.0\src\Fwk\Fwk.Bases\Blocks\Fwk.Logging\database\Logs.sqll

Almacén en archivos:

1- Generar una regla que apunte que informe que dicho tipo de log se almacenara un archivo

```
<add name="Rule2" events="Warning" target="Xml"  fileName="h:\logs\Logs.xml" />

O

<add name="Rule2" events="Warning" target="Text"  fileName="h:\logs\Logs.xml" />
```

Almacén en visor de sucesos de windows:

1- Generar una regla que apunte que informe que dicho tipo de log se almacenara un archivo

```
<add name="Rule3" events="Audit" target="WindowsEvent" />
```

Nota *por el momento no se puede definir el destino del log de windows y sera almacenado en windows log type = Application*

Apéndice B: Crear log cambiando de prefijo en el nombre de archivo

Si por algún motivo nos interesa efectuar logs de manera configurada pero queremos tambien que dentro del código se pueda variar el prefijo y ruta del archivo destino, lo podemos hacer utilizando otra sobrecarga del metodo Log:

```
Log(Event ev, string path, string fileNamePrefix)
```

path es la ruta donde se almacenara. Si es null o string.Empty por defecto se utiliza la ruta preconfigurada.-

fileNamePrefix es el prefijo que se antepone al nombre configurado en la regla.-

Ejemlo:

```
string prefix =  DateFunction.Get_Year_Mont_Day_String(System.DateTime.Now,'-');
StaticLogger.Log(ev,@"c:\",prefix);
```

Donde ***DateFunction.Get_Year_Mont_Day_String*** es una funcion de las HelperFunctions que retorna Retorna la fecha en formato YYYY[sep]MM[sep]dd