Constantino Sánchez Ballesteros
Dpto. Informática de *Sgrin, S.A. – Grupo CADIC*

Gestión de proveedores sobre el visor Web

Esta nueva *release* ha cambiado la arquitectura interna del visor Web respecto a la manipulación de proveedores. Desaparece la propiedad *ViewerHandler* en favor de *ProviderStateInfo*. Ésta última representa una lista descriptiva de los proveedores que manipule el visor mediante la clase *WVProviderStateInfo*. *WVProviderStateInfo* tiene 4 propiedades importantes:

- *PrvInfo*: del tipo *LtnProviderInfo*. Permite declarar la apertura de un proveedor.
- *Filter*: del tipo *LtnFilter*. Será el filtro que se utilice para definir los elementos que se quieren visualizar o consultar al proveedor.
- *Enabled*: del tipo *bool*. Indica si el proveedor será visible (dibujable) en el visor Web.
- *OptimizeForWMS*: esta propiedad se utiliza exclusivamente para optimizar ploteos sobre proveedores OGC que acceden a servicios WMS (Web Map Service). Si se establece a *true*, internamente se realizan *ploteos* de una única ventana de visualización en lugar de un *ploteo* por cada imagen que se muestra en el visor. Para otros tipos de proveedor no es necesario utilizar esta propiedad.

Con este nuevo soporte, cuando se crea un nuevo *WVProviderStateInfo* realmente no se crea físicamente un proveedor, sino que simplemente se declara para su posterior uso. Internamente el visor Web tiene implementado un *Pool* que va a permitir un uso eficiente y escalable de los proveedores sin necesidad de gestionar concurrencia por parte del programador de aplicaciones.

WVProviderStateInfo contiene varios métodos estáticos para manipular físicamente un proveedor del *Pool*:

- *WVProviderStateInfo.GetProvider(WVProviderStateInfo pInfo):* retorna un *LtnProvider* en base a una instancia de *WVProviderStateInfo*.
- *WVProviderStateInfo.ReleaseProvider(LtnProvider prv)*: devuelve al *Pool* un proveedor previamente obtenido con el método *GetProvider*.

Nota: Es importante recalcar que cada vez que se obtenga un proveedor del *Pool* es conveniente retornárselo después de su uso para su posterior reutilización, así ganaremos en optimización de recursos y en velocidad.

Veamos un ejemplo de cómo inicializar un visor Web con esta nueva arquitectura:

```
LtnProviderInfo prvInfo =
LtnProviderInfo.FromString("NAME=HTTP|LOCATION=localhost|ALIAS=prv|USER=usr|PWD=pwd");
WVProviderStateInfo psi = new WVProviderStateInfo(prvInfo, new LtnFilter(), true);
```



```
LtnProvider prv = WVProviderStateInfo.GetProvider(psi);
If (prv!=null)
{
      try
      {
            LtnPoint3D pMin,pMax;
            LtnSources sources = new LtnSources();
            sources.Add(prv);
            sources.Bounds(out pMin, out pMax);

            LtnWebViewer1.ProviderStateInfo = new WVProviderStateInfo[] {psi};
            LtnWebViewer1.DisplaySettings = new LtnDisplaySettings();
            LtnWebViewer1.SetBounds(pMin, pMax);
            LtnWebViewer1.SetViewWindow(pMin, pMax);
      }
      finally
      {
            WVProviderStateInfo.ReleaseProvider(prv);
      }
}
```

Primero se crea un descriptor de acceso *WVProviderStateInfo* al proveedor. El siguiente paso es obtener físicamente el proveedor en base al descriptor para calcular *bounds* y ventana de visualización inicial del visor web. Finalmente se asignan al visor los parámetros calculados y se devuelve el proveedor al *Pool*.

Consideraciones a tener en cuenta

Con esta nueva arquitectura hay que tener en cuenta un par de detalles importantes a la hora de manipular proveedores del lado del servidor:

1. Si se quiere manipular el filtro de un proveedor, se manipulará el filtro asignado al descriptor *WVProviderStateInfo* correspondiente en lugar del proporcionado por el propio proveedor. Internamente el proveedor tiene asignado el puntero del filtro al de su descriptor.

2. Si se quiere activar o desactivar un proveedor, se hará sobre la propiedad *Enabled* del descriptor *WVProviderStateInfo* en lugar de la proporcionada por el proveedor.

Aumento de rendimiento con dominios múltiples

Otra característica implementada en el visor Web es la gestión de múltiples dominios a la hora de realizar peticiones de imágenes. La especificación del protocolo HTTP 1.1 obliga a que no haya más de 2 conexiones concurrentes hacia un mismo dominio. Esto es un problema ya que el navegador encola peticiones de 2 en 2, echando por tierra la paralelización de peticiones y, por consiguiente, disminuyendo el rendimiento del visor.
Existen formas de saltarse esta limitación, normalmente requiriendo de ayuda del usuario para cambiar ciertos aspectos de la configuración del sistema operativo o del propio navegador.

LatinoObjects se salta esta limitación automáticamente y sin intervención del usuario utilizando la gestión de dominios múltiples. ¿Cómo? El visor Web tiene una propiedad *DomainNames* que permite asignar una lista de *string* con los nombres de dominio que se van a utilizar.

Para que todo funcione correctamente, cada uno de estos dominios tiene que apuntar a un mismo servidor. Para ello será preciso que el proveedor de servicios ajuste el DNS de cada dominio para que apunte a uno concreto.

Imaginemos que tenemos nuestra aplicación alojada bajo el dominio principal www.miweb.com. Puesto que vamos a usar esta nueva característica, damos de alta los siguientes dominios que apuntarán cada uno de ellos a www.miweb.com:

http://server01.miweb.com
http://server02.miweb.com
http://server03.miweb.com
http://server04.miweb.com

Asignamos esta lista de dominios a la propiedad *DomainNames* del visor Web de la aplicación y asunto arreglado. Habremos logrado tener concurrentes 8 peticiones de imágenes en lugar de 2.

