5. Engine Extensions

Nunit Test Engine usa un plugin de arquitectura que permite a los usuarios y a las terceras empresas añadir funcionalidades al mismo engine. Su modelo extensible define un número de puntos extensibles que permiten añadir extensiones. Esta página aporta información genetal para aplicar a todos los tipos de extensiones que podrías querer escribir. Las páginas individuales para cada tipo de extensión permiten añadir detalles específicos. Para una detallada descripción de a extensibilidad de la arquitectura del engine, [visita esta página](https://github.com/nunit/docs/wiki/Engine-Extensibility).

5.1 Extensión de atributo

Cada extensión es implementada por una clase con características específicas.

* Tiene un constructor por defecto con el que NUnit puede crear una instancia
* Implementa algunas interfaces que varían acorde a la extensión particular
* Está marcado con el “ExtensionAttribute” de NUnit que lo reconoce como una extensión.

El código de una extensión típica podría verse así:

[Extension]

public class MyExtension : ISomeInterface // Depending on the extension point

{

// Your code here

}

5.2 Propiedades del atributo

Todas estas propiedades son opcionales:

* **Ruta:** Es una String que identifica el punto de extensión en el cual se aplica. Es raramente necesitado desde que NUnit suele deducir el tipo de extensión basada a cual interfaz está implementada.
* **Descripción:** Es una descripción opcional de lo que la extensión hace.
* **Permitido:** Un boolean que indica si la extensión está permitida. Suele estar definida por defecto en true. La opción es usada por las extensiones avanzadas con funcionalidad que está encendida o apagada dependiendo de la introducción del usuario.
* **EngineVersion:** La versión mínima del engine que está soportada por la extensión. Aunque sea opcional, deberías usar esta propiedad si tu extensión no funciona en todas las versiones. Si no lo usas y tu extensión requiere servicios que no está presentes, podría producirse alguna excepción o causar otros errores. Las versiones 3.4 o superiores comprueban esta propiedad.

5.3 ExtensionAttribute

Usando solo ExtensionAttribute, NUnit podría tener que crear instancias de cada extensión en orden para preguntar cosas como “Qué extensión de archivo soportas”. Esto podría significar cargar muchas librerías posiblemente no necesitadas.

5.4 ExtensionPropertyAttribute

ExtensionPropertyAttribute evade este problema. La extensión propia de NUnit para cargar proyectos de NUnit es un buen ejemplo:

[Extension]

[ExtensionProperty("FileExtension", ".nunit")]

public class NUnitProjectLoader : IProjectLoader

{

...

}

Con su uso, NUnit permite posponer la carga de la extensión hasta que el usuario utiliza un archivo de extensión .unit

5.5 Tipos de extensiones

Desde la versión 3.4, NUnit Engine soporta 4 tipos de extensiones. A continuación se detallan páginas individuales para cada tipo específico con el fin de ofrecer detalles de cada una de ellas.

* [Project Loaders](https://github.com/nunit/docs/wiki/Project-Loaders)
* [Result Writers](https://github.com/nunit/docs/wiki/Result-Writers)
* [Framework Drivers](https://github.com/nunit/docs/wiki/Framework-Drivers)
* [Event Listeners](https://github.com/nunit/docs/wiki/Event-Listeners)