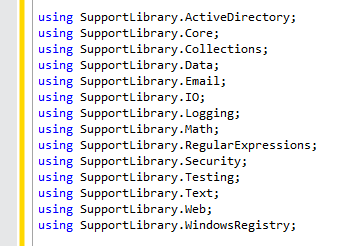
SupportLibrary v1.0.0.12

Les dejo un pantallazo de las funcionalidades principales dentro de la SupportLibrary, con ejemplos de uso, listos para copiar y pegar ☺

Como regla general a las clases que acceden a recursos externos se les provee constructores, haciéndolas instanciables. Gracias a esto, es que se puede proveer además de sus respectivas interfaces, facilitando su reemplazo con mocks y stubs al hacer Unit Testing. Por otro lado, a las clases que solamente parsean o procesan un input para generar una salida, sin recurrir a recursos externos, se las hace estáticas para una mayor facilidad de uso.

Namespaces



Contents

[Namespace ActiveDirectory 3](#_Toc494967298)

[Namespace Collections 5](#_Toc494967299)

[Namespace Core 7](#_Toc494967300)

[Namespace Data 13](#_Toc494967301)

[Namespace Email 15](#_Toc494967302)

[Namespace IO 16](#_Toc494967303)

[Namespace Logging 18](#_Toc494967304)

[Namespace Logging.TraceListeners 23](#_Toc494967305)

[Namespace Math 24](#_Toc494967306)

[Namespace RegularExpressions 25](#_Toc494967307)

[Namespace RegularExpressions.PasswordValidator 27](#_Toc494967308)

[Namespace Security 28](#_Toc494967309)

[Namespace Testing 29](#_Toc494967310)

[Namespace Text 33](#_Toc494967311)

[Namespace Web 34](#_Toc494967312)

[Namespace WindowsRegistry 36](#_Toc494967313)

# Namespace ActiveDirectory

* Clase **ActiveDirectoryHelper**

|  |  |
| --- | --- |
| Utilidad | Leer propiedades del Active Directory para el usuario dado. |
| Aplicación inmediata | ChargeTransfer (analizar uso tmb en Control de Permisos). |

* Métodos **GetProperty, GetProperty <TResult>**
* Se puede consultar mediante un String con el nombre de la propiedad en el ActiveDirectory (todas) o en forma tipificada usando la sobrecarga con el enum **DirectoryEntryProperty** (propiedades más comunes).
* El enum **DirectoryEntryProperty** incluye: CompleteName, FirstName, LastName, DisplayName, AccountName, PrincipalName, TelephoneNumber, Email, HomeDirectory, HomeDrive, Department1, Title (posición), etc.
* Parámetro RevertImpersonation (true/false).

string name1 = new ActiveDirectoryHelper().GetProperty<string>("username", DirectoryEntryProperty.CompleteName); // USERNAME IT

string name2 = new ActiveDirectoryHelper().GetProperty<string>("username", "cn"); // USERNAME IT

string tel = new ActiveDirectoryHelper().GetProperty<string>("username", DirectoryEntryProperty.TelephoneNumber); // 11-1000-2000

string email = new ActiveDirectoryHelper().GetProperty<string>("username", DirectoryEntryProperty.Email); // user@domain.name.com

string home = new ActiveDirectoryHelper().GetProperty<string>("username", DirectoryEntryProperty.HomeDirectory); // \\SERVER\USERNAME

* Método **FindDirectoryEntry**
* Si se van a consultar varias propiedades por usuario, se puede traer su nodo en un único acceso al Active Directory y consultar luego sobre el objeto en memoria.
* Se puede usar en tándem con **DirectoryEntryHelper** para facilitar el acceso sobre la entidad DirectoryEntry.

DirectoryEntry directoryEntry = new ActiveDirectoryHelper().FindDirectoryEntry("username")

* Clase **DirectoryEntryHelper**

|  |  |
| --- | --- |
| Utilidad | Proporciona métodos estáticos para acceder a las propiedades más comunes de un DirectoryEntry. |
| Aplicación inmediata | ChargeTransfer (analizar uso tmb en Control de Permisos). |

* Métodos **GetProperty, GetProperty <TResult>**

DirectoryEntry directoryEntry = new ActiveDirectoryHelper().FindDirectoryEntry("username")

DirectoryEntryHelper.GetProperty<string>(directoryEntry, DirectoryEntryProperty.PrincipalName);

DirectoryEntryHelper.GetProperty<string>(directoryEntry, DirectoryEntryProperty.AccountName);

DirectoryEntryHelper.GetProperty<string>(directoryEntry, DirectoryEntryProperty.Title);

# Namespace Collections

* Clase **ArrayEx**

|  |  |
| --- | --- |
| Utilidad | Realizar operaciones sobre Arrays como si fueran estructuras dinámicas. |
| Aplicación inmediata | Cualquier aplicación que haga uso extensivo de arrays. |

string[] array = new string[] { "1", "2", "3" };

List<string> list = new List<string>() { "4", "5", "6" };

Dictionary<string, string> dic = new Dictionary<string, string>();

* Métodos **Add <T>, Add IEnumerable <T>**
* Redimensiona el Array recibido y le agrega el nuevo valor/secuencia al final.
* Como los Arrays por default se pasan por valor, para que pueda modificarse, debe invocarse con el parámetro **ref** por delante.

ArrayEx.Add(ref array, "4"); // add valor

ArrayEx.Add(ref array, new string[] { "4", "5", "6" }); // add array

ArrayEx.Add(ref array, list); // add lista

ArrayEx.Add(ref array, dic.Keys.AsEnumerable()); // add de cualquier clase que implemente IEnumerable

* Método **Concatenate <T>, Concatenate IEnumerable <T>**
* Crea un nuevo Array como la unión de los valores del Array recibido más el nuevo valor al final.
* El Array recibido no se modifica, y el nuevo Array se retorna como resultado.

array = ArrayEx.Concatenate(array, "4"); // concatenar valor

array = ArrayEx.Concatenate(array, new string[] { "4", "5", "6" }); // concatenar array

array = ArrayEx.Concatenate(array, list); // concatenar lista

array = ArrayEx.Concatenate(array, dic.Keys.AsEnumerable()); // concatenar cualquier clase que implemente IEnumerable

* Método **RemoveAt <T>**
* Elimina el valor de una posición dada. El Index de la posición a eliminar empieza en cero.
* Como los Arrays por default se pasan por valor, para que pueda modificarse, debe invocarse con el parámetro **ref** por delante.

array = ArrayEx.RemoveAt(ref array, 1); // "1", "2", "3" -> "1", "3"

* Método **RemoveAll <T>**
* Elimina todos los valores que cumplan con una condición dada. Retorna la cantidad de elementos eliminados.
* Como los Arrays por default se pasan por valor, para que pueda modificarse, debe invocarse con el parámetro **ref** por delante.

int count = ArrayEx.RemoveAll(ref array, a => a == "2"); // "1", "2", "3" -> "1", "3", count = 1

* Clase **Extensions**
* Permiten usar el método Concatenate<T> directamente desde una instancia de Array.

array = array.Concatenate("4"); // concatenar valor

array = array.Concatenate(new string[] { "4", "5", "6" }); // concatenar array

array = array.Concatenate(list); // concatenar lista

array = array.Concatenate(dic.Keys.AsEnumerable()); // concatenar cualquier clase que implemente IEnumerable

# Namespace Core

* Clase **Cast**

|  |  |
| --- | --- |
| Utilidad | Funcionalidad para castear objetos entre distintos tipos. |
| Aplicación inmediata | Sistema de Facturación. Cualquier proyecto. |

* Permite castear instancias de objetos de un tipo “source” a nuevas instancias de otro tipo “dest”.
* Ideal para aplicar sobre una misma clase que se serializó bajo diferentes namespaces al pasar por una capa de web services.
* Se mapean vía reflexion las propiedades de ambas clases que coincidan en su nombre. Las propiedades de la clase destino que no tengan coincidencia en la clase origen, se establecen con el valor por default según su tipo de datos (int = 0, datetime = 1/1/0001, tipos complejos = null).
* El último parámetro permite elegir si mapear solo las propiedades del primer nivel (más rápido) o si mapear también las de objetos internos, anidados por composición (valor por default).
* Método **To<TResult>**
* La sintaxis es bastante flexible, ejemplos:

// PersonEx -> Person, mapear también objetos anidados.

Person person = Cast.To<Person>(personEx, true);

// Ídem anterior.

Person person = Cast.To<Person>(personEx);

// Sobrecarga <source, dest> con una instancia de la clase destino. El tipo a retornar se infiere del de la clase destino.

Person person = Cast.To(personEx, new Person(), true);

// Ídem anterior.

Person person = Cast.To(personEx, new Person());

* Ejemplo de uso:

// arrange

Person inputPerson = BuildPerson();

// act

PersonEx outputPersonEx = Cast.To<PersonEx>(inputPerson, false);

// assert

Assert.AreEqual(inputPerson.Id, outputPersonEx.Id, "Assert 01");

Assert.AreEqual(inputPerson.FirstName, outputPersonEx.FirstName, "Assert 02");

Assert.AreEqual(inputPerson.LastName, outputPersonEx.LastName, "Assert 03");

Assert.AreEqual(inputPerson.Age, outputPersonEx.Age, "Assert 04");

Assert.AreEqual(inputPerson.Tall, outputPersonEx.Tall, "Assert 05");

Assert.AreEqual(default(float), outputPersonEx.Weight, "Assert 06");

Assert.AreEqual(null, outputPersonEx.Address, "Assert 07");

* Método **CollectionTo<T, TResult>**
* Para castear sobre listas de objetos, misma sintaxis que con el método To<TResult>, ejemplos:

// List<Person> -> List<PersonEx>

List<PersonEx> persons = Cast.CollectionTo<Person, PersonEx>(personsEx, false);

// Sobrecarga con instancia de la clase destino.

List<PersonEx> persons = Cast.CollectionTo(personsEx, new PersonEx(), false);

* Ejemplo de uso:

// arrange

List<Person> inputPersons = new List<Person>();

inputPersons.Add(BuildPerson());

inputPersons.Add(BuildPerson());

// act

List<PersonEx> outputPersons = Cast.CollectionTo(inputPersons, new PersonEx(), true);

// assert

for (int i = 0; i < outputPersons.Count; i++)

{

Assert.AreEqual(inputPersons[i].Id, outputPersons[i].Id, "Assert Id");

Assert.AreEqual(inputPersons[i].FirstName, outputPersons[i].FirstName, "Assert FirstName");

Assert.AreEqual(inputPersons[i].LastName, outputPersons[i].LastName, "Assert LastName");

Assert.AreEqual(inputPersons[i].Age, outputPersons[i].Age, "Assert Age");

Assert.AreEqual(inputPersons[i].Tall, outputPersons[i].Tall, "Assert Tall");

Assert.AreEqual(default(float), outputPersons[i].Weight, "Assert Weight");

Assert.AreEqual(inputPersons[i].Address.StreetName, outputPersons[i].Address.StreetName, "Assert StreetName");

Assert.AreEqual(inputPersons[i].Address.StreetNumber, outputPersons[i].Address.StreetNumber, "Assert StreetNumber");

Assert.AreEqual(inputPersons[i].Address.City, outputPersons[i].Address.City, "Assert City");

Assert.AreEqual(inputPersons[i].Address.State, outputPersons[i].Address.State, "Assert State");

Assert.AreEqual(inputPersons[i].Address.ZipCode, outputPersons[i].Address.ZipCode, "Assert ZipCode");

}

* Clase **Clone**

|  |  |
| --- | --- |
| Utilidad | Funcionalidad para clonar objetos y colecciones. |
| Aplicación inmediata | Cualquier proyecto. |

* Permite hacer clonado superficial/profundo sobre instancias de objetos o colecciones.
* Las clases sobre las que se realizarán las copias deben implementar la interfaz SupportLibrary.Core.ICloneableExtended.
* Métodos **Clone(**bool**)**, **DeepClone()**, **ShallowClone()**

Person person = new Person();  
  
Person shallowClone = person.Clone(false); // shallow copy

Person deepClone = person.Clone(true); // deep copy

* Clase **DateTimeEx**

|  |  |
| --- | --- |
| Utilidad | Operaciones varias que involucren Fechas. |
| Aplicación inmediata | Cualquier proyecto. |

* Métodos **LastDayOfMonth** & **FirstDayOfNextMonth**
* Calculan fechas como desplazamientos desde una fecha de inicio dada.
* Si no se especifica un DateTime como parámetro, se calcular desde hoy (DateTime.Today).

DateTimeEx.LastDayOfMonth(new DateTime(2017, 09, 19)); // 2017-09-30  
DateTimeEx.FirstDayOfNextMonth(new DateTime(2017, 09, 19)); // 2017-10-01

* Clase **Debug**

|  |  |
| --- | --- |
| Utilidad | Funcionalidad para facilitar tareas de debug. |
| Aplicación inmediata | Cualquier proyecto. |

* Método **GetVarName**
* Obtiene el nombre de cualquier variable.
* Símil a operador nameof() de C# 6.0 (Visual Studio 2015).
* Utiliza como input expresiones de Linq apuntando a un delegado. Sintaxis: () => var
* Ideal si se quiere guardar mayor detalle en algún archivo de log sin hardcodear el nombre de la(s) variable(s).

public void Example(string blNumber, string cntrNumber)

{

Logger.Log(String.Format("blNumber = {0}", blNumber));

}

vs

public void Example(string blNumber, string cntrNumber)

{

Logger.Log(String.Format("{0} = {1}", Debug.GetVarName( () => blNumber ), blNumber));

}

* Idem al generar errores en tiempo de ejecución:

public void Example(string param1, string param2)

{

if (param1 == null)

{

throw new ArgumentNullException("param1", "The parameter 'param1' can't be null.");

}

}

vs

public void Example(string param1, string param2)

{

if (param1 == null)

{

string varName = Debug.GetVarName<string>(() => param1); // "param1"

throw new ArgumentNullException(varName, String.Format("The parameter '{0}' can't be null.", varName));

}

}

* Método **LoggedExecution**
* Loguea y executa el método y los parámetros dados.

[under development]

* Método **TimedExecution**
* Permite medir el tiempo transcurrido en la ejecución de un método dado.

Func<string> func = () => TestMethod("Test", 123);

TimeSpan timeSpan = Debug.TimedExecution(func);

* Clase **Serializer**

|  |  |
| --- | --- |
| Utilidad | Funcionalidad para facilitar tareas de serialización/deserialización. |
| Aplicación inmediata | Cualquier proyecto. |

* Métodos **SerializeToBinary** & **DeserializeFromBinary**

* Serializa objetos hacia/desde formato binario usando el serializador BinaryFormatter.

byte[] serializedPerson = Serializer.SerializeToBinary<Person>(person);

Person deserializedPerson = Serializer.DeserializeFromBinary<Person>(serializedPerson);

* Métodos **SerializeToXml** & **DeserializeFromBinary**
* Serializa objetos hacia/desde formato binario usando los serializadores DataContractSerializer o XmlSerializer.

string serializedPerson = Serializer.SerializeToXml<Person>(person, XmlSerialization.DataContract);

string serializedPerson = Serializer.SerializeToXml<Person>(person, XmlSerialization.XmlSerializer);

Person deserializedPerson = Serializer.DeserializeFromXml<Person>(serializedPerson, XmlSerialization.DataContract);

Person deserializedPerson = Serializer.DeserializeFromXml<Person>(serializedPerson, XmlSerialization.XmlSerializer);

* Métodos **SerializeToFile** & **DeserializeFromFile**
* Serializa objetos hacia/desde formato binario usando los serializadores BinaryFormatter, DataContractSerializer o XmlSerializer.

Serializer.SerializeToFile<Person>(person, filePath, fileName, Serialization.BinaryFormatter);

Serializer.SerializeToFile<Person>(person, filePath, fileName, Serialization.DataContract);

Serializer.SerializeToFile<Person>(person, filePath, fileName, Serialization.XmlSerializer);

Person deserializedPerson = Serializer.DeserializeFromFile<Person>(filePath, fileName, Serialization.BinaryFormatter);

Person deserializedPerson = Serializer.DeserializeFromFile<Person>(filePath, fileName, Serialization.DataContract);

Person deserializedPerson = Serializer.DeserializeFromFile<Person>(filePath, fileName, Serialization.XmlSerializer);

* Clases **Extensions**

|  |  |
| --- | --- |
| Utilidad | Permitir el uso de las funcionalidades del namespace SupportLibrary.Core directamente sobre instancias de objeto. |
| Aplicación inmediata | Cualquier proyecto. |

* Permiten usar el método **CastTo** directamente desde cualquier instancia de objeto.
* Permiten usar el método **CastCollectionTo** directamente desde cualquier colección que implemente IList<T>.
* Permiten usar el método **GetValue** y **GetDescription** directamente desde cualquier instancia de Enum.
* Permiten usar los métodos **SerializeToBinary** & **DeserializeFromBinary** directamente desde instancias de objetos.
* Permiten usar los métodos **SerializeToXml** & **DeserializeFromXml** directamente desde cualquier instancia de objeto.
* Permiten usar el método **SerializeToFile** directamente desde cualquier instancia de objeto.
* Permiten usar los métodos **LastDayOfMonth** & **FirstDayOfNextMonth** directamente desde cualquier instancia de objeto del tipo DateTime.

# Namespace Data

* Clase **DataHelper**

|  |  |
| --- | --- |
| Utilidad | Basado en la funcionalidad del SqlHelper v1.0.0.12, clase SQLHelper.Utils.Utils.  Manejo de Nulls en ResultSets. Conversión de listas y entidades a DataTable. |
| Aplicación inmediata | Todos los proyectos. |

* Método **ReplaceDbNull <T>**
* Símil funcionalidad del SqlHelper v1.0.0.12
* Métodos **ThrowIfDbNull <T>**
* Símil funcionalidad del SqlHelper v1.0.0.12
* Se agrega sobrecarga para poder elegir la Excepción generada (en lugar de generar siempre una ApplicationException).

DataHelper.ThrowIfDbNull<string>(drWithNulls["StringColumn"], new Exception("Mensaje de error 1."));

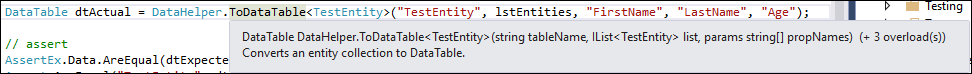
DataHelper.ThrowIfDbNull<string>(drWithNulls["StringColumn"], new ArgumentException("Mensaje de error 2."));

DataHelper.ThrowIfDbNull<string>(drWithNulls["StringColumn"], new ArgumentNullException("column1", "Mensaje de error 3."));

DataHelper.ThrowIfDbNull<string>(drWithNulls["StringColumn"], typeof(ArgumentException), "Mensaje de error 2."));

* Métodos **ToDataTable <T>**
* Misma funcionalidad del SqlHelper v1.0.0.12
* Se agrega sobrecarga para poder setear el nombre del DataTable generado.

DataTable dtActual = DataHelper.ToDataTable<TestEntity>("TableTestEntity", lstEntities, "FirstName", "LastName", "Age");



* Clase **Extensions**
* Permiten usar los métodos ReplaceDbNull, ThrowIfDbNull directamente desde un DataRow.
* Permiten usar el método ToDataTable directamente desde cualquier colección que implemente IList<T>.
* Clase **SqlHelper**

|  |  |
| --- | --- |
| Utilidad | Basado en la funcionalidad del SqlHelper v1.0.0.12, clase SQLHelper.SQL\_Helper.  **Reescritura del código desde cero** para mayor simplicidad. Cambios en la firma de los constructores para mayor consistencia.  Ejecución de Stored Procedures. Con o sin resultados (en DataTable o DataSet). Con o sin logging. |
| Aplicación inmediata | Todos los proyectos. |

* Método **ExecuteStored**
* Símil funcionalidad del SqlHelper v1.0.0.12. Siempre devuelve el valor retornado por el sp.

int returnValue1 = new SqlHelper("SupportLibrary").ExecuteStored("sp\_SupportLibrary\_GetData", "param1");

* Método **ExecuteStoredLog**
* Ídem al anterior, con activando logueo de la ejecución.

int returnValue1 = new SqlHelper("SupportLibrary").ExecuteStoredLog("sp\_SupportLibrary\_GetData", "param1");

* Método **GetDataTable**
* Símil funcionalidad del SqlHelper v1.0.0.12.

DataTable dtActual = new SqlHelper("SupportLibrary").GetDataTable("sp\_SupportLibrary\_GetLogs", dateTimeStart, tuples.Count);

* Método **GetDataSet**
* Símil funcionalidad del SqlHelper v1.0.0.12.

DataSet dsActual = new SqlHelper("SupportLibrary").GetDataSet("sp\_SupportLibrary\_GetData", dateTimeStart, tuples.Count);

# Namespace Email

* Clase **EmailHelper**

|  |  |
| --- | --- |
| Utilidad | Proporciona métodos estáticos para facilitar validaciones, envíos de email y tareas relacionadas. |
| Aplicación inmediata | SendMail v3. |

* Métodos **SplitAddresses**
* Parsea un string con direcciones de email separadas por coma o punto. Verifica que todas las direcciones estén bien formadas y las devuelve en una lista en forma individual.

List<string> lstEmail = EmailHelper.SplitAddresses("asd@test.com, qwerty@test.com; zxcv@test.com");

List<string> lstEmail = EmailHelper.SplitAddresses("asd@test.com # qwerty@test.com |zxcv@test.com", new char[] { '|', '#' });

* Método **IsValidEmailAddress**

bool validity = EmailHelper.IsValidEmailAddress("test@asd.com");

# Namespace IO

* Clase **FileHelper**

|  |  |
| --- | --- |
| Utilidad | Facilitar tareas de lectura/escritura de archivos. |
| Aplicación inmediata | Cualquier proyecto que lea o guarde archivos o reportes a disco. |

* Sobrecargas con parámetro RevertImpersonation (true/false).
* Métodos **ReadBinary**
* Lee archivos binarios.

byte[] fileContent = new FileHelper().ReadBinary(filePath, fileName);

* Métodos **ReadText**
* Lee archivos de texto.

string fileContent = new FileHelper().ReadText(filePath, fileName);

* Métodos **SaveBinary**
* Guardar archivos binarios.

MemoryStream memoryStream = new MemoryStream();

byte[] inputContent = memoryStream.ToArray(); // any secuence, dont matter what

new FileHelper().SaveBinary(filePath, fileName, inputContent);

* Métodos **SaveText**
* Guardar archivos de texto.
* Modos Create (sobreescribe si ya existe) o Append (agregar al final).

string inputContent = "This is a test string. This is another test string.";

new FileHelper().SaveText(filePath, fileName, inputContent); // Create Mode

new FileHelper().SaveText(filePath, fileName, TextFileMode.Create, inputContent, false);

new FileHelper().SaveText(filePath, fileName, TextFileMode.Append, inputContent1, false);

* Métodos **DeleteOldFiles**
* Elimina archivos antiguos del path elegido.
* Permite filtrar tanto por antigüedad como por patrones en el nombre de archivo.

new FileHelper().DeleteOldFiles(filePath, "\*.txt", 7); // All text files in path, from 7 days back

new FileHelper().DeleteOldFiles(filePath, "File4.\*", 7); // All files named File4 in path, from 7 days back

new FileHelper().DeleteOldFiles(filePath, null, 0); // All files in path

# Namespace Logging

* Clase **EventLogHelper**

|  |  |
| --- | --- |
| Utilidad | Facilitar lectura/escritura en el EventLog de Windows. |
| Aplicación inmediata | Todos los proyectos tipo Servicio de Windows (acostumbran guardar la información importante allí). |

* El seteo de LogName y LogSource se puede hacer tanto a nivel Constructor como en forma individual al llamar al método de Log.
* Se pueden loguear tanto Mensajes (entrada tipo information, warning o error) como Excepciones (genera una entrada del tipo error).
* Métodos **Log**

private const string LOG\_NAME = "SupportLibrary";

private const string LOG\_SOURCE = "SupportLibraryLogSource";

EventLogHelper eventLogHelper = new EventLogHelper(LOG\_NAME, LOG\_SOURCE);

eventLogHelper.Log(EntryType.Info, "An Info message.");

eventLogHelper.Log(EntryType.Warning, "A Warning message.");

eventLogHelper.Log(EntryType.Error, "An Error message.");

eventLogHelper.Log(new ArgumentException("The parameter is invalid."));

new EventLogHelper(LOG\_NAME, LOG\_SOURCE).Log(EntryType.Info, "An Info message.");

new EventLogHelper(LOG\_NAME, LOG\_SOURCE).Log(EntryType.Warning, "A Warning message.");

new EventLogHelper(LOG\_NAME, LOG\_SOURCE).Log(EntryType.Error, "An Error message.");

new EventLogHelper().Log(LOG\_NAME, LOG\_SOURCE, EntryType.Info, "An Info message.");

new EventLogHelper().Log(LOG\_NAME, LOG\_SOURCE, EntryType.Warning, "A Warning message.");

new EventLogHelper().Log(LOG\_NAME, LOG\_SOURCE, EntryType.Error, "An Error message.");

* Clase **TextLogHelper**

|  |  |
| --- | --- |
| Utilidad | Facilitar lectura/escritura de logs en archivos de texto. |
| Aplicación inmediata | Cualquier proyecto que necesite guardar logs simples a disco. |

* Pensada como una alternativa ‘liviana’ al Apache log4net y la funcionalidad de logging de la MS Enterprise Library.
* Sin configuraciones complejas, integrada, simple y fácil de usar.
* Permite loguear tanto mensajes como excepciones.
* El constructor recibe el path y nombre de archivo. Opcionalmente se puede seleccionar el Template de base para las entradas de log y un parámetro RevertImpersonation (true/false).
* Métodos **Log**
* Guarda una nueva entrada en el archive de log.
* Los formatos de Template para las entradas de log son 3: Minimal, Standard y Custom.
* **TEMPLATE MINIMAL**

Template header

title {datetime} ({level}) ->

footer

Ejemplo 1

TextLogHelper logger = new TextLogHelper(filePath, fileName, EntryTemplate.Minimal);

logger.Log(EntryType.Info, "Infomation entry with minimal profile enabled.");

logger.Log(EntryType.Warning, "Warning entry with minimal profile enabled.");

logger.Log(EntryType.Error, "Error entry with minimal profile enabled.");

Output

2016/09/23 05:54:18 (INFO) -> Infomation entry with minimal profile enabled.

2016/09/23 05:54:18 (WARNING) -> Warning entry with minimal profile enabled.

2016/09/23 05:54:19 (ERROR) -> Error entry with minimal profile enabled.

Ejemplo 2

...

textLogger = new TextLogHelper(@"C:\Log\", "SendMail.log", EntryTemplate.Minimal);

LogMessage(item, moduleName, ActionTaken.Deleted, "It is an email from an user at domain.com.");

private void LogMessage(Item item, string moduleName, ActionTaken actionTaken, string actionDetail)

{

StringBuilder sb = new StringBuilder(Environment.NewLine);

sb.AppendLine(" - FromAddress : " + item.fromAddress);

sb.AppendLine(" - Subject : " + item.Subject);

sb.AppendLine(" - Received : " + item.DateTimeReceived.ToString("yyyy/MM/dd hh:mm:ss"));

sb.AppendLine(" - Module : " + moduleName);

sb.AppendLine(" - Action : " + actionTaken.ToString().ToUpper());

sb.AppendLine(" - Detail : " + actionDetail);

textLogger.Log(EntryType.Info, sb.ToString());

}

Output 2

2016/09/23 07:06:16 (INFO) ->

- FromAddress : account @domain.name.com

- Subject : Autorización ref. 123IAR0444054

- Received : 2016/09/16 11:10:38

- Module : DeleteMails

- Action : DELETED

- Detail : It is an email from an user at domain.name.com.

2016/09/23 07:12:16 (INFO) ->

- FromAddress : account@domain.name.com

- Subject : Nueva Transaccion (N) Nro 123IAR0444431

- Received : 2016/09/16 11:04:33

- Module : ForwardMailsByAddress

- Action : FORWARDED

- Detail : LogId: 15148. Forwarded to: account@domain.name.com

* **TEMPLATE STANDARD**

Template header ----------------------------------------

title {datetime} - {level}

footer ----------------------------------------

Ejemplo

TextLogHelper logger = new TextLogHelper(filePath, fileName, EntryTemplate.Standard);

logger.Log(EntryType.Info, infoMessage);

logger.Log(EntryType.Warning, warningMessage);

logger.Log(EntryType.Error, errorMessage);

Output

----------------------------------------

2016/09/23 05:56:24 - INFO

Infomation entry with standard profile enabled.

----------------------------------------

----------------------------------------

2016/09/23 05:56:24 - WARNING

Warning entry with standard profile enabled.

----------------------------------------

----------------------------------------

2016/09/23 05:56:24 - ERROR

Error entry with standard profile enabled.

----------------------------------------

* **TEMPLATE CUSTOM**

En modo CUSTOM, el campo Title acepta los siguiente placeholders de reemplazo:

{id} Se reemplaza en las entradas sucesivas por números incrementales (1, 2, 3, 4… ), según formateo.

{level} Se reemplaza por el tipo de entrada de log (INFO, WARNING, ERROR), según formateo.

{datetime} Se reemplaza por la fecha/hora actual, según formateo.

Estos, a su vez, pueden ser formateados según las expresiones de String.Format().

Más info en: <https://msdn.microsoft.com/es-es/library/system.string.format.aspx/>

Ejemplos {id:0000} --> '0001'

{level,8} --> ' INFO'

{level,-6} --> 'INFO '

{datetime:dd-MM-yyyy} --> '31-12-2016'

Template header Según propiedad ‘CustomHeader’.

title Según propiedad ‘CustomTitle’.

footer Según propiedad ‘CustomFooter’.

Ejemplo

TextLogHelper loggerCustom = new TextLogHelper(filePath, fileName, EntryTemplate.Custom);

loggerCustom.SetTitle("{datetime:yyyy-MM-dd hh:mm:ss} - {level,-12} ({id:00})");

loggerCustom.SetHeader('-', 40).SetFooter('-', 40);

loggerCustom.Log(EntryType.Info, infoMessage);

loggerCustom.Log(EntryType.Warning, warningMessage);

loggerCustom.Log(EntryType.Error, errorMessage);

Output

----------------------------------------

2016-09-23 05:56:24 - INFO (01)

Infomation entry with custom profile enabled.

----------------------------------------

----------------------------------------

2016-09-23 05:56:24 - WARNING (02)

Warning entry with custom profile enabled.

----------------------------------------

----------------------------------------

2016-09-23 05:56:24 - ERROR (03)

Error entry with custom profile enabled.

----------------------------------------

# Namespace Logging.TraceListeners

* Clases **CustomConsoleTraceListener** & **CustomEventLogTraceListener**

|  |  |
| --- | --- |
| Utilidad | Formatos personalizados de listeners para eventos de Trace. |
| Aplicación inmediata | Todos tipo de proyectos. |

* TraceListeners que vuelcan mensajes a la Consola y el EventLog de Windows, respectivamente.
* Permiten adjuntar información adicional, además del mensaje volcado en el evento de Trace, por ejemplo la fecha & hora, nombre del método o nombre de clase donde se generó el evento.
* El constructor de la clase recibe un enum que identifica la información adicional a incorporar al mensaje de Trace. La sobrecarga sin parámetros 0 la opción TraceOption.None implican no agregar ningúna información adicional.
* Métodos **Write** & **WriteLine**

Trace.Listeners.Clear();

// output: { mensaje }

Trace.Listeners.Add(new CustomConsoleTraceListener(TraceOption.None));

// output: 05/10/2017 11:00 AM - { mensaje }  
Trace.Listeners.Add(new CustomConsoleTraceListener(TraceOption.DateTimeOnly));

// output: VepSync.OnTick() – { mensaje }

EventLog eventLog = new EventLog("VepSyncService", ".", "VepSyncLogSource");

CustomEventLogTraceListener eventLogTraceListener = new CustomEventLogTraceListener(eventLog, TraceOption.ClassAndMethod);

Trace.Listeners.Add(eventLogTraceListener);

# Namespace Math

* Clase **MathEx**

|  |  |
| --- | --- |
| Utilidad | Proporciona métodos estáticos para facilitar operaciones matemáticas. |
| Aplicación inmediata | Sistema de cálculos. |

* Métodos **Truncate**
* Trunca un número float, double o decimal a un máximo de posiciones decimales.

float floatValue = MathEx.Truncate(-1234.5678F, 2); // float -1234.56F

double doubleValue = MathEx.Truncate(1234.5678D, 0); // double 1234D

decimal decimalValue = MathEx.Truncate(1234.5678M, 2); // decimal 1234.56M

# Namespace RegularExpressions

* Clase **RegexHelper**

|  |  |
| --- | --- |
| Utilidad | Proporciona métodos estáticos para facilitar operaciones con expresiones regulares. |
| Aplicación inmediata | ImportCntrControl, ChargeRules, ChargeException. Cualquier proyecto. |

* Clase **Business**, fields **BILL\_OF\_LADDING, BOOKING, CONTAINER, SHIPPING\_INSTRUCTION**
* Proveen expresiones regulares de referencia, listas para usar en validaciones de objetos de negocio.

string bkPattern = RegexHelper.Business.BOOKING; // @"^((AR|IAR|123AR|123IAR)[0-9]{7}|^((EBK)[0-9]{8}))$"

string blPattern = RegexHelper.Business.BILL\_OF\_LADDING; // @"^MMC([a-zA-Z]{2}[0-9]{7}|[a-zA-Z]{3}[0-9]{6})$"

* Clase **Business**, métodos **IsValidBillOfLaddingNumber, IsValidBookingNumber, IsValidContainerNumber, IsValidShippingInstruction**
* Permiten validar fácilmente un objeto de negocio.

bool result1 = RegexHelper.Business.IsValidBillOfLaddingNumber("MMCAA1234567");

bool result2 = RegexHelper.Business.IsValidBookingNumber("123AR1234567");

bool result3 = RegexHelper.Business.IsValidContainerNumber("AAAA1234567");

bool result4 = RegexHelper.Business.IsValidShippingInstruction("123456789");

* Clase **Business**, método **IdentifyDocument**
* Permite identificar rápidamente que a tipo de objeto de negocio corresponde un string (input de usuario, etc).
* El enum **BusinessDocument** incluye: Unknown, BillOfLadding, Booking, Container y ShippingInstruction.

BusinessDocument businessDocument = RegexHelper.Business.IdentifyDocument("123AR1234567"); // booking

* Clase **Miscellaneous**, fields **EMAIL, IP, NUMERIC, URL, CUIT\_NUMBER, y otros**
* Proveen expresiones regulares de referencia, listas para usar en validaciones varias.

string url = RegexHelper.Miscellaneous.URL; // @"/^(https?:\/\/)?([\da-z\.-]+)\.([a-z\.]{2,6})([\/\w \?=.-]\*)\*\/?$/";

* Clase **Miscellaneous**, método **IsNumeric**
* Permite validar fácilmente si un texto está compuesto exclusivamente por números.

bool result = RegexHelper.Miscellaneous.IsNumeric ("1234567890"); // true

* Clase **Miscellaneous**, método **IsValidCuitNumber**
* Permite validar fácilmente si un texto es un número de CUIT válido.

bool result = RegexHelper.Miscellaneous.IsValidCuitNumber("30-52671272-9"); // true

bool result = RegexHelper.Miscellaneous.IsValidCuitNumber("30526712729"); // false, dashes missing

bool result = RegexHelper.Miscellaneous.IsValidCuitNumber("11-11111111-1"); // false, wrong CUIT

bool result = RegexHelper.Miscellaneous.IsValidCuitNumber("30526712729", false); // true

bool result = RegexHelper.Miscellaneous.IsValidCuitNumber("30-52671272-9", false); // false, included dashes

# Namespace RegularExpressions.PasswordValidator

* Clase **PasswordValidator**

|  |  |
| --- | --- |
| Utilidad | Permite facilitar las validaciones de passwords. |
| Aplicación inmediata | Validación de datos del cliente |

* Métodos **Validate** & **ValidateVerbose**
* Chequea que el password actual cumpla con todas las validaciones configuradas.
* El método ValidateVerbose devuelve una entidad ValidationResult con información detallada.

Se incluye: resultado general de todas las validaciones, lista de las validaciones que fallaron y un mensaje descriptivo de estas.

PasswordValidator pv = new PasswordValidator("abmldq3421!");

pv.Validations.SetMinLenght(8); // longitud minima 8

pv.Validations.SetMaxLenght(50); // longitud maxima 50

pv.Validations.SetMinNumericChars(1); // al menos un número

pv.Validations.SetMinSpecialChars(1); // al menos un carácter especial (@#$! etc)

pv.Validations.SetMaxConsecutiveChars(5); // hasta 5 veces el mismo carácter en forma consecutiva

pv.Validations.SetMaxIncrementalChars(3); // hasta 3 caracteres en secuencia creciente (123, abc, etc)

pv.Validations.SetMaxDecrementalChars(3); // hasta 3 caracteres en secuencia decreciente (543, zyx, etc)

bool result = pv.Validate();

ValidationResult resultVerbose = pv.ValidateVerbose();

# Namespace Security

* Clase **Cryptography**

|  |  |
| --- | --- |
| Utilidad | Proporciona métodos estáticos para facilitar el cálculo de Hashes. |
| Aplicación inmediata | Cualquier proyecto. En especial los que manejen cuentas de usuario. |

* Métodos **ComputeHashSHA1** y **ComputeHashMD5**.
* Permite realizar cálculos de hashes en una única línea de código.

string input = "This is a test string.";  
string hash1 = Cryptography.ComputeHashSHA1(input);  
string hash2 = Cryptography.ComputeHashMD5(input);

# Namespace Testing

* Clase **AssertEx**

|  |  |
| --- | --- |
| Utilidad | Proporciona métodos estáticos para facilitar operaciones de Test Unitario. |
| Aplicación inmediata | Cualquier proyecto. |

* Pensada como un complemento a la clase Assert de MSTest (el sufijo ‘Ex’ es por ‘Extensions’).
* Permite varias de las aserciones “avanzadas” de xUnit, sin el uso de esta librería externa.
* Clase **Collection**, métodos **IsEmpty, IsNotEmpty**

AssertEx.Collections.IsEmpty(new int[0]); // pass

AssertEx.Collections.IsEmpty(new ArrayList()); // pass

AssertEx.Collections.IsEmpty(new Hashtable()); // pass

AssertEx.Collections.IsEmpty(new int[] { 1, 2, 3 }); // fails

AssertEx.Collections.IsNotEmpty(new int[] { 1, 2, 3 }); // pass

AssertEx.Collections.IsNotEmpty(new int[0]); // fails

* Clase **Condition**, métodos **Greater, GreaterOrEqual, Less, LessOrEqual**
* Opera entre cualesquiera Tipos que implementen IComparable (todos los tipos primitivos, por ejemplo).
* Si se chequea por ‘Greater’, el primer argumento se espera que sea el mayor y el segundo argumento el menor.
* Si se chequea por ‘Less’, el primer argumento se espera que sea el menor y el segundo argumento el mayor.

AssertEx.Condition.Greater(20, 10); // pass

AssertEx.Condition.Greater<int>(20, 10); // pass

AssertEx.Condition.Greater(10, 20); // fails

AssertEx.Condition.Greater(10, 10); // fails

AssertEx.Condition.GreaterOrEqual(1.0, 1.0); // pass

AssertEx.Condition.GreaterOrEqual(1.0, 1.1); // fails

* Clase **Data**, métodos **AreEqual**
* Para hacer aserciones por igualdad entre DataRows, DataTables y DataSets.
* Útil en aplicaciones que se usen DataTable o DataSets para transportar datos.

// arrange

DataTable dataTable1 = new DataTable("Table1");

dataTable1.Columns.Add("Name", Type.GetType("System.String"));

dataTable1.Columns.Add("Age", Type.GetType("System.Byte"));

dataTable1.Rows.Add("Pablo", 20);

dataTable1.Rows.Add("Maria", 30);

DataTable dataTable2 = new DataTable("Table2");

dataTable2.Columns.Add("Name", Type.GetType("System.String"));

dataTable2.Columns.Add("Age", Type.GetType("System.Byte"));

dataTable2.Rows.Add("Pablo", 20);

dataTable2.Rows.Add("Maria", 30);

// assert

AssertEx.Data.AreEqual(dataTable1, dataTable2, "Assert 01");

* Aserciones para verificar que se lance una excepción, métodos del framework MSTest.

1. El atributo [ExpectedException()] puede resultar inadecuado en determinadas (¡muchas!) situaciones, con este método no se pueden hacer aserciones sobre los datos de la excepción generada.

No se puede por ejemplo: ArgumentException.ParamName == "1234", Exception.Message.Contains("1234"), Exception.InnerException.GetType() is xxx

1. Usar la opción del Try… Catch() salda los faltantes del método anterior, pero solo permite testear de a una aserción a la vez, lo que puede resultar en extremo tedioso.

[TestMethod, ExpectedException(typeof(ArgumentNullException))]

public void TestMethod()

{

try

{

string userName = null; // arrange

MailRepository.Get(userName); // act, null -> must throw an ArgumentNullException!

Assert.Fail("No exception was thrown."); // assert

}

catch (ArgumentNullException) { /\* assertion pass \*/ }

catch (Exception) { Assert.Fail("Different exception type was thrown."); }

}

* Clase **Exceptions**, método **Throws <TException>**
* Inspirado en su homónimo de xUnit, verifica que el método ejecutado lance una excepción, y que esta sea del tipo TException especificado.
* Especialmente útil en los test unitarios de lógica inversa, en los que se espera que determinada lógica falle y lance una excepción.
* Permite mayor flexibilidad que los métodos estándar del framework MSTest.
* Recibe un delegado al método que debe generar excepción.

Ejemplo 1.

[TestMethod]

public void TestMethod()

{

// arrange

List<TestEntity> lstEntities = new List<TestEntity>()

{

new TestEntity() { Id = 1, FirstName = "Pablo", LastName = "Gutierrez", Age = 20, Tall = 1.75F },

new TestEntity() { Id = 2, FirstName = "Maria", LastName = "Lopez", Age = 30, Tall = 1.65F },

new TestEntity() { Id = 3, FirstName = "Carlos", LastName = "Gonzalez", Age = 40, Tall = 1.70F }

};

Action action1 = () => DataHelper.ToDataTable(lstEntities, "FirstName", "InvalidPropertyName", "Age");

Action action2 = () => DataHelper.ToDataTable(lstEntities, "FirstName", null, "Age");

// act & assert

ArgumentException exception1 = AssertEx.Exceptions.Throws<ArgumentException>(action1, "Assert 01");

ArgumentNullException exception2 = AssertEx.Exceptions.Throws<ArgumentNullException>(action2, "Assert 02");

Assert.AreEqual("propNames", exception1.ParamName, "Assert 03");

Assert.AreEqual("propNames", exception2.ParamName, "Assert 04");

}

Ejemplo 2.

[TestMethod]

public void TestMethod()

{

// arrange

string textValid = "asd@test.com, qwerty@test.com"; // valid text (with default delimiters)

string textInvalid1 = "asd@test.com @ qwerty@test.com"; // invalid text (with default delimiters)

string textInvalid2 = "asd@test.com . qwerty@test.com"; // invalid text (with default delimiters)

Action action1 = () => EmailHelper.SplitAddresses(textValid, '\*');

Action action2 = () => EmailHelper.SplitAddresses(textInvalid1);

Action action3 = () => EmailHelper.SplitAddresses(textInvalid1, '@');

Action action4 = () => EmailHelper.SplitAddresses(textInvalid2);

Action action5 = () => EmailHelper.SplitAddresses(textInvalid2, '.');

// act

ArgumentException exception1 = AssertEx.Exceptions.Throws<ArgumentException>(action1, "Assert 01");

ArgumentException exception2 = AssertEx.Exceptions.Throws<ArgumentException>(action2, "Assert 02");

ArgumentException exception3 = AssertEx.Exceptions.Throws<ArgumentException>(action3, "Assert 03");

ArgumentException exception4 = AssertEx.Exceptions.Throws<ArgumentException>(action4, "Assert 04");

ArgumentException exception5 = AssertEx.Exceptions.Throws<ArgumentException>(action5, "Assert 05");

}

* Clase **Text**, métodos **IsEmpty, IsNotEmpty, AreEqualIgnoringCase**
* Inspirado en su homónimo de xUnit, verifica que el método ejecutado lance una excepción, y que esta sea del tipo TException especificado.

AssertEx.Text.IsEmpty(""); // pass

AssertEx.Text.IsEmpty("This is a test string."); // fails

AssertEx.Text.IsEmpty((string)null); // fails

AssertEx.Text.IsNotEmpty("This is a test string."); // pass

AssertEx.Text.IsNotEmpty(""); // fails

AssertEx.Text.AreEqualIgnoringCase("name", "NAME"); // pass

AssertEx.Text.AreEqualIgnoringCase("MMC", "MMC String"); // fails

# Namespace Text

* Clase **Extensions**

|  |  |
| --- | --- |
| Utilidad | Proporciona métodos estáticos para facilitar operaciones con Strings. |
| Aplicación inmediata | Cualquier proyecto. |

* Pequeños agregados al operar con instancias de strings.
* El método **IsNullOrEmpty** permite aplicar el chequeo directamente sobre la instancia. No solo como método estático de la clase String.
* Se agrega el opuesto: **IsNotNullOrEmpty**, que en determinadas circunstancias puede ser más claro usarlo así, que negar al **IsNullOrEmpty**.
* El método **IsNumeric** permite evaluar si un string está compuesto únicamente por números.
* El método **SafeSusbtring** permite extraer una subcadena, pero sin preocuparse por posiciones fuera de rango: de incluirse posiciones inválidas estas se ignoran.
* El método **SubstringByRange** permite extraer una subcadena expecificando un rango desde-hasta. Equivalente a Subtring(0, endIndex - startIndex).
* El método **Truncate** permite cortar desde el inicio de una cadena hasta un largo máximo. Equivalente a Substring (0, lenght), pero sin tener que chequear el largo máximo para evitar excepción.
* El método **Split** permite separar una cadena usando una **secuencia de strings** como separador.

string testString = "This is a test string. This is another test string.";

bool result = testString.IsNumeric() // false

bool result = testString.IsNullOrEmpty() // false

bool result = testString.IsNotNullOrEmpty() // true

string result = testString.SafeSubstring(0, 1000) // "This is a test string. This is another test string."

string result = testString.SubstringByRange(23, 38) // "This is another"

string result = testString.Truncate(22); // "This is a test string."

string result = testString.Truncate(100); // "This is a test string. This is another test string."

string[] result = testString.Split("a", "e", "i", "o", "u");

# Namespace Web

* Clase **SessionHelper**

|  |  |
| --- | --- |
| Utilidad | Facilitar el acceso a los valores de Sesion de manera tipada. |
| Aplicación inmediata | Cualquier proyecto. |

* Métodos **Retrieve<T>,** **RetrieveWithMessage<T>** y **RetrieveWithResourceKey<T>**.
* Extrae un valor de sesión, genera error en caso de no existir el valor.

// si no existe genera una ArgumentException, con el texto "Invalid Session. Please re-start the application".

Token token = new SessionHelper().Retrieve<Token>(LOGIN\_TOKEN);

// si no existe genera una ArgumentException, con el texto del segundo parámetro.  
Token token = new SessionHelper().RetrieveWithMessage<Token>(LOGIN\_TOKEN, "No se encontró el valor en sesión.");

// si no existe genera una ArgumentException, tomando el texto desde los LocalResources con la ResourceKey dada.

Token token = new SessionHelper().RetrieveWithRecourceKey<Token>(LOGIN\_TOKEN, "\_InvalidSession");

* Métodos **TryRetrieve<T>**.
* Intenta extraer un valor de sesión, retorna el resultado de la operación y una referencia al valor encontrado (o null en caso que no exista).

public bool ValidateLoginToken()

{

try

{

Token token = new Token();

return new SessionHelper().TryRetrieve<Token>(LOGIN\_TOKEN, ref token);

}

catch (Exception) { throw; }

}

* Clase **FileDownloader**

|  |  |
| --- | --- |
| Utilidad | Facilitar la descarga archivos desde una página web. |
| Aplicación inmediata | Cualquier proyecto. |

* Métodos **Download**
* Manda a descargar un archivo mediante el objeto HttpContext.Response recibido.
* Entre los formatos de archivo soportados están: texto, cvs, pdf, word (doc y docx), excel (xls y xlsx), jpg y png.
* Parámetros:

El primer parámetro se utiliza para establecer en las cabeceras del Response el ContentType (MIME) del archivo a descargar.

El segundo parámetro indica el nombre de archivo sugerido al cliente por el browser.

El tercer parámetro lleva el contenido del archivo.

FileDownloader fileDownloader1 = new FileDownloader(HttpContext.Current); // constructor con HttpContext

fileDownloader1.Download(FileType.TEXT, fileName, fileContent); // archivo de texto (string)

FileDownloader fileDownloader2 = new FileDownloader(HttpContext.Current.Response); // constructor con HttpResponse

fileDownloader2.Download(FileType.BINARY, filename, fileBuffer); // archivo binario (byte[])

* Clase **Extensions**

|  |  |
| --- | --- |
| Utilidad | Facilitar la descarga archivos desde una página web. |
| Aplicación inmediata | Cualquier proyecto. |

* Métodos **DownloadFile**
* Permiten usar el método **FileDownloader.Download** directamente desde el objeto **HttpContext**.

HttpContext.Current.DownloadFile(FileType.PDF, fileName, fileBuffer);

# Namespace WindowsRegistry

* Clase **RegistryHelper**

|  |  |
| --- | --- |
| Utilidad | Proporciona métodos estáticos para facilitar la lectura/escritura en el Registro de Windows. |
| Aplicación inmediata | Todos los proyectos. |

* Inspirado en el RegistryHelper ya existente.
* Por default, direcciona a las subclaves de ‘LocalMachine\SOFTWARE\ApplicationName\’)
* También proporciona sobrecargas para el acceso completo de las ramas LocalMachine y CurrentUser del Registro.
* Agrega soporte para la mayoría de los tipos de datos permitidos en el Registro de Windows, a saber: String, DWORD (int), QWORD (long) y Binary (byte[]).
* Métodos **GetKeyValue, GetKeyValue <TResult>**

byte[] binaryData = new RegistryHelper().GetKeyValue<byte[]>("SupportLibrary", "TestValueBinary");

int integerData = new RegistryHelper().GetKeyValue<int>("SupportLibrary", "TestValueInteger");

long longData = new RegistryHelper().GetKeyValue<long>("SupportLibrary", "TestValueLong");

string stringData = new RegistryHelper().GetKeyValue<string>("SupportLibrary", "TestValueString");

string stringData = new RegistryHelper().GetKeyValue<string>(RegistryHive.LocalMachine, "SOFTWARE\\ApplicationName\\", "SupportLibrary",

"TestValueString");

* Métodos **SetKeyValue**

new RegistryHelper().SetKeyValue("SupportLibrary", RegistryValueType.Binary, "TestValueBinary", BitConverter.GetBytes(1234));

new RegistryHelper().SetKeyValue("SupportLibrary", RegistryValueType.Integer, "TestValueInteger", 1234);

new RegistryHelper().SetKeyValue("SupportLibrary", RegistryValueType.Long, "TestValueLong", 1234);

new RegistryHelper().SetKeyValue("SupportLibrary", RegistryValueType.String, "TestValueString", "Test");

* Métodos **GetKeyValues y SetKeyValues**
* Inspirado en la clase RegistryHelper del proyecto AfipConector.
* Para aplicaciones que leen y escriben frecuentemente de a bloques de clave/valores el registro.
* En el Get, recibe el nombre de la clave a leer y retorna un Dictionary<string, object> con todos los pares <nombreValor, datoValor> encontrados.
* En el Set, recibe el nombre de la clave a escribir y un Dictionary<string, object> con todos los pares <nombreValor, datoValor> a guardar. Como datoValor (object) solo se permiten los tipos byte[], int, long y string.

Dictionary<string, object> keyValues = new Dictionary<string, object>();

keyValues.Add("TestValueBinary", BitConverter.GetBytes(1234));

keyValues.Add("TestValueInteger", 1234);

keyValues.Add("TestValueLong", 1234L);

keyValues.Add("TestValueString", "Test");

new RegistryHelper().SetKeyValues("SupportLibrary", keyValues);

Dictionary<string, object> keyValuesResult = new RegistryHelper().GetKeyValues("SupportLibrary");