Tabla de contenidos

JSON

1.1 Definición

JSON (**JavaScript Object Notation** - **Notación de Objetos de JavaScript**) es un formato ligero de intercambio de datos.

Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo.

Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999.

JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros.

1.2 Estructura

Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.

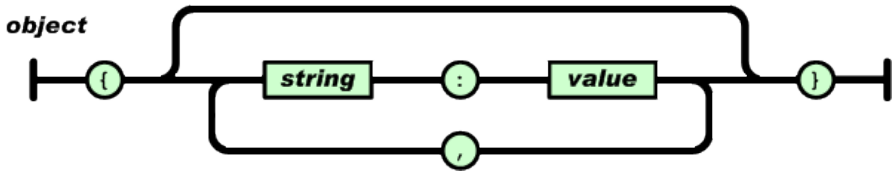
Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas, etc.

JSON son estructuras universales y virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación.

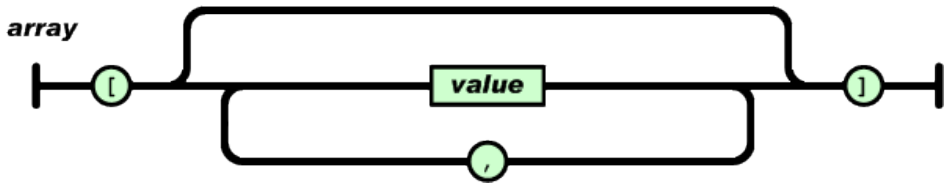
1.3 Tipos de datos

En JSON, se presentan de estas formas:

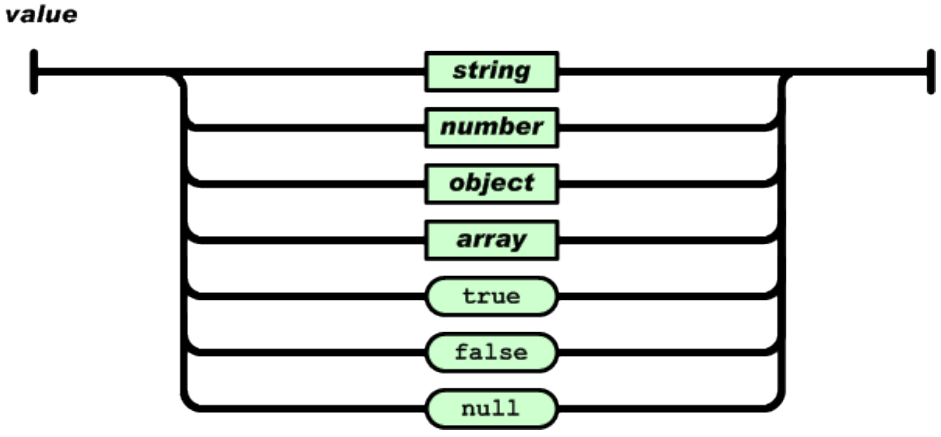
Objetos: Un objeto es un conjunto sin ordenar de pares clave-valor. Comienza por "{" y termina con "}".Cada nombre estará seguido por ":", los pares clave-valor estarán separados por ",".



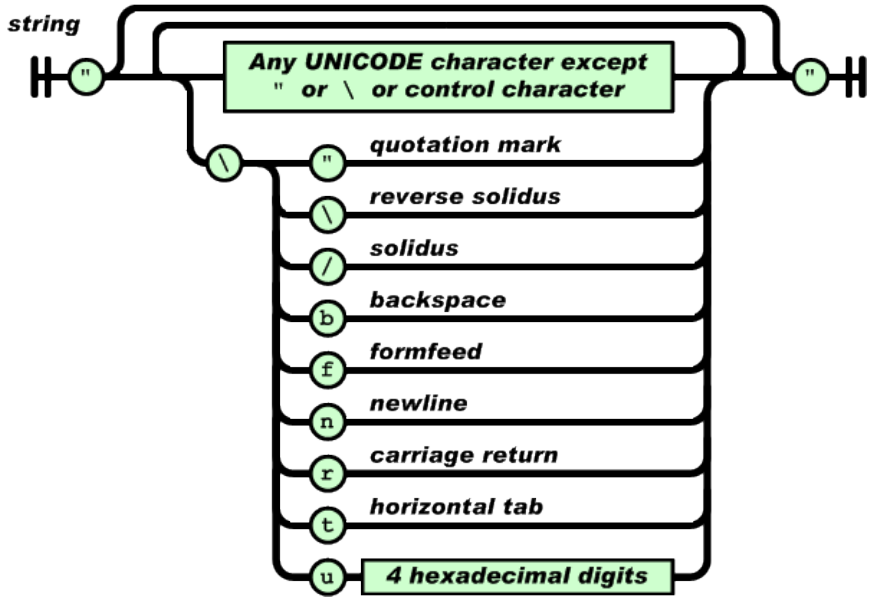
Array: Un array es una colección de valores. Comienza por "[" y finaliza con "]". Los valores se separan por ",".



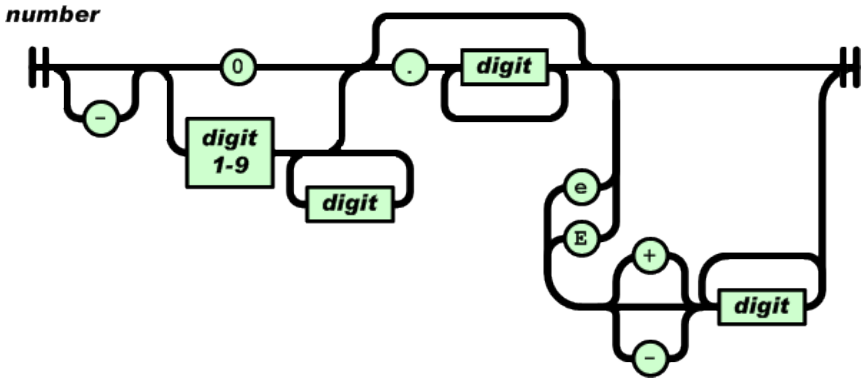
Valor: Un valor puede ser una cadena de caracteres con comillas doble, un número, true, false, null, un objeto o un array. Estas estructuras pueden anidarse:



Cadena: Una cadena de caracteres es una colección de cero o más caracteres Unicode, encerrados entre comillas dobles, usando barras divisorias invertidas como escape. Un carácter está representado por una cadena de caracteres de un único carácter. Una cadena de carateres es parecida a una cadena de caracteres C o Java.



Número: Un número es similar a un número C o Java, excepto que no se usan los formatos octales y hexadecimales.



1.4 Esquemas y ejemplos

Los objetos JSON se identifican entre llaves, un objeto puede ser en nuestro caso una fruta o una verdura.

Ejemplo 1:

```
"{ 'NombreFruta':'Manzana' ,'Cantidad':20 }"
```

En un Json puedes incluir arrays, para ellos el contenido del array debe ir entre corchetes []:

Ejemplo 2:

```
"{'Frutas': [{ 'NombreFruta':'Manzana' , 'cantidad':10 }, {
'NombreFruta':'Pera' , 'cantidad':20 }, {
'NombreFruta':'Naranja' , 'cantidad':30 }]}"
```

Json.NET

Es un marco JSON alto rendimiento popular para .NET

Beneficios y Características

1. Serializador JSON flexible para convertir entre objetos .NET y JSON
2. Alto rendimiento: más rápido que .NET incorporado en serializadores JSON
3. Convertir JSON a XML
4. Soporta 2 .NET, .NET 3.5, .NET 4, .NET 4.5, Silverlight, Windows Phone y Windows 8 Store

Json.NET surgió de los proyectos que estaba trabajando a finales de 2005 la participación de JavaScript, AJAX, y .NET. En ese momento no había bibliotecas para trabajar con JavaScript en .NET, así que hice mi propia.

Comenzó como un par de métodos estáticos para escapar de las cadenas de JavaScript, Json.NET evolucionó como se agregaron características. Para añadir soporte para la lectura de un JSON refactor importante se requería, y Json.NET se dividió en las tres clases principales que todavía utiliza hoy: JsonReader, JsonWriter y JsonSerializer.

2.1 Instalación json.net (NuGet)

La instalación de json.net se lleva a cabo desde visual studio posicionándose en el menú **herramientas>administrador de paquetes de**

biblioteca>Administrador de paquetes NuGet para solución como se muestra en la Figura 2.1

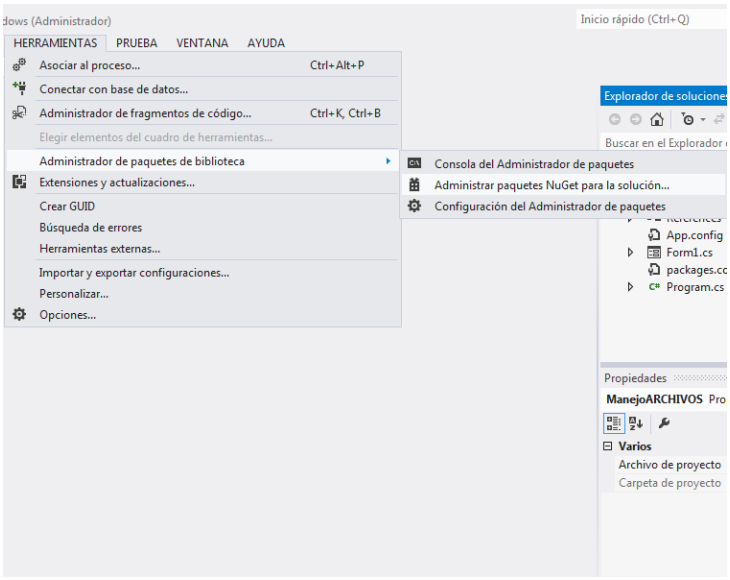


Figura 2.1 Instalación de NuGet

En el administrador de paquetes se buscar json.net y se selecciona la instalación como se muestra en la figura 2.2

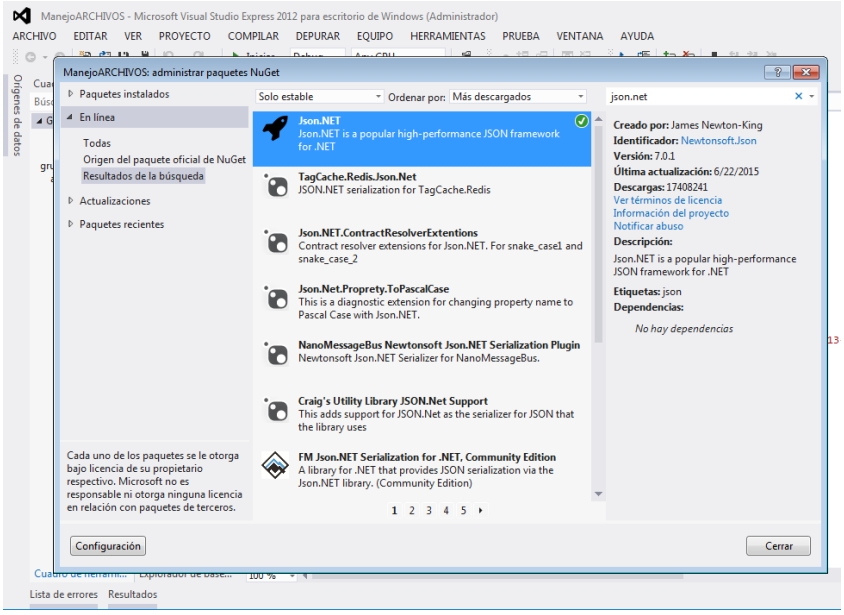


Figura 2.1 Json.NET

2.2 Serialización DataSet a JSON

Para realizar una serialización con la JSON.NET a partir de un DataSet se utiliza:

JsonConvert.SerializeObject

También se puede guardar los datos de la serialización en un documento .json como la siguiente fragmento de código:

```
string json = JsonConvert.SerializeObject(DataSset);

System.IO.File.WriteAllText(@"C:\archivo.json", json);
```

La instrucción generara un resultado como este:


```
{"Productos":[{"ID":"1","Nombre":"Pantalla Plasma","Precio":"3000"},{"ID":"2","Nombre":"Reproductor DVD","Precio":"1000"},{"ID":"3","Nombre":"Mesa redonda","Precio":"10000"},{"ID":"4","Nombre":"Cama XLL","Precio":"8000"}]}
```


2.3 Serialización Colección a JSON

Con la serialización de una colección podemos crear colecciones completas y convertirlas en un documento JSON, para entender esta funcionalidad que tiene la serialización se puede realizar lo siguiente:

```
// Se crea la lista

List<string> videogames = new List<string>{"Starcraft","Halo","Zelda"};

string json = JsonConvert.SerializeObject(videogames);

// Se guardan los datos en un archivo json

System.IO.File.WriteAllText(@"C:\videogames.json", json);
```

2.4 Serialización diccionario a JSON

Con la serialización de un diccionario podemos crear un archivo o información en formato JSON, para entender esta funcionalidad que tiene la serialización se puede realizar lo siguiente:

```
// Se un diccionario de datos en este caso se crean puntos
Dictionary<string, int> points = new Dictionary<string, int>{{ "James", 9001 },{ "Jo", 3474 },{ "Jess", 11926 }};

// Se convierten los puntos mediante JsonConvert

string json = JsonConvert.SerializeObject(points);

// Se guardan los datos en un archivo json

System.IO.File.WriteAllText(@"C:\points.json", json);
```

2.5 Serialización de objetos a JSON

Con la serialización de un objeto podemos crear un archivo o información en formato JSON a partir de un objeto definido con c#, para entender esta funcionalidad que tiene la serialización se puede realizar lo siguiente:

Si tenemos a disposición una clase Account como por ejemplo:

```
public class Account
   {
      public string Email { get; set; }
      public bool Active { get; set; }
      public DateTime CreatedDate { get; set; }
      public IList<string> Roles { get; set; }
}
```

Podemos crear objetos tipo Account que posteriormente serán usados para generar un archivo JSON.

```
Account account = new Account{ Email = "james@example.com",Active = true,CreatedDate = new DateTime(2013, 1, 20, 0, 0, 0, DateTimeKind.Utc),
         Roles = new List<string>{"User","Admin"}};
```

```
string json = JsonConvert.SerializeObject(account);
        System.IO.File.WriteAllText(@"C:\account.json", json);
```

2.6 Deserialización JSON a objeto

Con la Deserialización de un archivo json podemos crear un Objeto o información a partir del formato JSON, para entender esta funcionalidad utilizamos la clase Account y utilizamos un archivo serializado .json:

```
Account account =
JsonConvert.DeserializeObject<Account>(System.IO.File.ReadAllText(@"C:\ac
count.json"));
MessageBox.Show(account.Email);
```

2.7 Deserialización JSON a colección

Con la Deserialización de un archivo json podemos crear una colección o información a partir del formato JSON o una variable con información json, para entender esta funcionalidad se puede usar la siguiente instrucción:

```
string json = @"['Starcraft','Halo','Legend of Zelda']";
List<string> videogames =
JsonConvert.DeserializeObject<List<string>>(json);
MessageBox.Show(string.Join(", ", videogames));
```

2.8 Deserialización JSON a DataSet

Con la Deserialización de un archivo json podemos almacenar la información a un DataSet a partir del formato JSON o una variable con información json, para entender esta funcionalidad se puede usar la siguiente instrucción:

```
DS =
JsonConvert.DeserializeObject<DataSet>(System.IO.File.ReadAllText(@"C:\
productos.json"));
dataGridView1.DataSource = DS.Tables[0];
```

Referencias:

http://www.newtonsoft.com/json/help/html/SerializeDataSet.htm

http://www.newtonsoft.com/json/help/html/Overload_Newtonsoft_Json_JsonConvert_DeserializeObject.htm

http://www.newtonsoft.com/json/help/html/SerializeWithJsonSerializerToFile.htm

http://www.newtonsoft.com/json/help/html/DeserializeWithJsonSerializerFromFile.htm

http://www.newtonsoft.com/json/help/html/SerializeCollection.htm

http://www.newtonsoft.com/json/help/html/SerializeDictionary.htm

http://www.newtonsoft.com/json/help/html/DeserializeObject.htm

http://www.newtonsoft.com/json/help/html/DeserializeCollection.htm

http://www.newtonsoft.com/json/help/html/DeserializeDictionary.htm