|  |  |
| --- | --- |
| Geo Parking | JSON  2014 |

El documento describe información JSON que servirá como capacitación en el desarrollo del producto Geo Parking.

# Control de la documentación

### Control de la Configuración.

|  |  |
| --- | --- |
| Título: | JSON |
| Referencia: | GeoP\_Proyecto\_JSON.docx |
| Autores: | Lucas Toneatto |
| Fecha: | 24/08/2014 |

### Histórico de Versiones.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versión | Fecha | Estado | Responsable | Cambios |
| 1.0 | 24/08/2014 | Aprobado | Lucas Toneatto[autor] |  |
|  |  |  |  |  |
|  |  |  |  |  |

Contenido

[Control de la documentación 2](#_Toc396658683)

[Control de la Configuración. 2](#_Toc396658684)

[Histórico de Versiones. 2](#_Toc396658685)

[Introducción 4](#_Toc396658686)

[Sintaxis JSON 4](#_Toc396658687)

[Objetos 4](#_Toc396658688)

[Array 4](#_Toc396658689)

[Herramientas de Desarrollo 4](#_Toc396658690)

[Formas de Generación de JSON 5](#_Toc396658691)

[Método ToJSON() con JavaScriptSerializer 5](#_Toc396658692)

[Serializacion utilizando JSON.Net 6](#_Toc396658693)

[JSON en Entity Framework 6](#_Toc396658694)

[Aplicación práctica en GeoParking 7](#_Toc396658695)

[JSON y JavaScript 9](#_Toc396658696)

[Referencias 10](#_Toc396658697)

# Introducción

JSON (JavaScript Object Notation) es un formato para el intercambios de datos, básicamente JSON describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos. JSON nació como una alternativa a XML, el fácil uso en JavaScript ha generado un gran número de seguidores de esta alternativa. Una de las mayores ventajas que tiene el uso de JSON es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías.

# Sintaxis JSON

JSON utiliza lo que se denomina una sintaxis de pares “nombre/valor”. Para asignar a un nombre un valor, se utilizan los “:”, lo cual sería comparable con el “=” de cualquier otro lenguaje.

***"Nombre”: "Geeky Theory"***

Los valores que podemos manejar en JSON son los siguientes:

1. Numero
2. String
3. Booleano
4. Array
5. Objeto
6. Null

## Objetos

Los objetos en JSON deben estar entre llaves (“{}”) como por ejemplo

***{ "NombreFruta":"Manzana" , "Cantidad":20 }***

## Array

El contenido de un array en JSON debe de ir entre corchetes (“[]”) como por ejemplo

**{**

**"Frutas": [**

**{"NombreFruta":"Manzana" , "cantidad":10 },**

**{"NombreFruta":"Pera" , "cantidad":20 },**

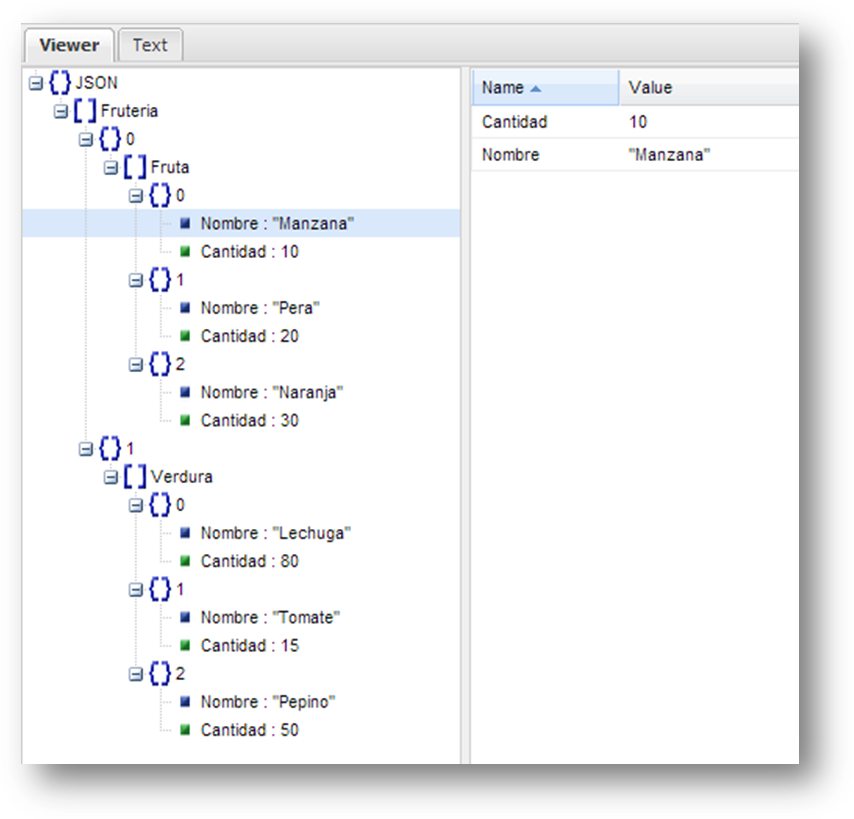
**{"NombreFruta":"Naranja" , "cantidad":30 }**

**]**

**}**

# Herramientas de Desarrollo

Existen herramientasonline que ayudan a visualizar mejor un JSON. Una de las mejores herramientas que he visto es la página [JSON Viewer](http://jsonviewer.stack.hu/). Si introducimos nuestro ejemplo observamos lo siguiente:

[](http://i2.wp.com/geekytheory.com/wp-content/uploads/2013/10/Captura11.png)

# Formas de Generación de JSON

### Método ToJSON() con JavaScriptSerializer

class DatosPersona

{

public string Nombre { get; set; }

public string Apellidos { get; set; }

public string TipoDocumento { get; set; }

public string Documento { get; set; }

*public string ToJSON()*

*{*

*System.Web.Script.Serialization.JavaScriptSerializerjsonSerializer*

*= new System.Web.Script.Serializaion.JavaScriptSerializer();*

*return jsonSerializer.Serialize(this);*

*}*

}

### Serializacion utilizando JSON.Net

class Persona {

public Guid Id { get; set; }

public string Nombre { get; set; }

public string Apellidos { get; set; }

public string FechaNacimiento { get; set; }

}

public void serializar{

var devjoker = new Persona() {

Id = Guid.NewGuid(),

Nombre = "Pedro",

Apellidos = "Herrarte",

FechaNacimiento = new DateTime (1975,01,13).ToString("d")

};

var json = Newtonsoft.Json.JsonConvert.SerializeObject(devjoker);

}

# JSON en Entity Framework

Todo lo explicado anteriormente es aplicable a clases simples o clases con relaciones simples; en esos casos la serialziacion y deserializacion se puede hacer de manera automática con JSON.Net.

Pero en el caso de Entity Framework cuando existen relaciones bidireccionales entre dos o mas clases, a la hora de serializar el objeto u entidad, se produce lo que se de denomina una excepción por existencia de referencia circular, lo cual JSON no lo maneja.

Es por ello que existen varias formas para solucionar esto, acá se detallaran a modo de ejemplo solo tres de ellas.

1. Clases simples para las vitas: consta de crear clases que solo contengan atributos simples que no generen la excepción de la referencia circular. Si bien es muy fácil de aplicar es tener una duplicación de información.
2. Exclusión de atributos: consta de marcar aquellos atributos que causan la excepción para que no sean serializados. Si bien es efectivo, se esta solucionando modificando el modelo para solucionar algo que es problema de la vista.
3. Proyección o método personalizado de socialización: consta de que cada clase genere la estructura de su JSON. Es similar a tener el método toString(), en este caso seria toJSON().

# Aplicación práctica en GeoParking

En esta sección se mostrara a modo de ejemplo como se trabaja la serializaron JSON de las entidades del proyecto. Aquí se mostrara la aplicación sobre la entidad “PlayaDeEstacionamiento”.

/// <summary>

/// retorna el objeto en formato JSON

/// </summary>

/// <returns>ObjetoJSON</returns>

public String ToJSONRepresentation()

{

StringBuilder sb = new StringBuilder();

JsonWriter jw = new JsonTextWriter(new StringWriter(sb));

jw.Formatting = Formatting.Indented;

//comienzo a escribrir el objeto

jw.WriteStartObject();

jw.WritePropertyName("Id");

jw.WriteValue(this.Id);

jw.WritePropertyName("Nombre");

jw.WriteValue(this.Nombre);

jw.WritePropertyName("Mail");

jw.WriteValue(this.Mail);

jw.WritePropertyName("Telefono");

jw.WriteValue(this.Telefono);

jw.WritePropertyName("TipoPlaya");

jw.WriteValue(this.TipoPlayaStr);

jw.WritePropertyName("Latitud");

jw.WriteValue(this.Direcciones[0].Latitud);

jw.WritePropertyName("Longitud");

jw.WriteValue(this.Direcciones[0].Longitud);

//DIRECCIONES

jw.WritePropertyName("Direcciones");

jw.WriteStartArray();

int i;

i = 0;

for (i = 0; i < this.Direcciones.Count; i++)

{

jw.WriteStartObject();

jw.WritePropertyName("Calle");

jw.WriteValue(Direcciones[i].Calle);

jw.WritePropertyName("Numero");

jw.WriteValue(Direcciones[i].Numero);

jw.WriteEndObject();

}

jw.WriteEndArray();

//SERVICIOS

jw.WritePropertyName("Servicios");

jw.WriteStartArray();

int j;

j = 0;

for (j = 0; j < this.Servicios.Count; j++)

{

jw.WriteStartObject();

jw.WritePropertyName("TipoVehiculo");

jw.WriteValue(Servicios[j].TipoVehiculoStr);

jw.WritePropertyName("Capacidad");

jw.WriteValue(Servicios[j].Capacidad);

jw.WriteEndObject();

}

jw.WriteEndArray();

//HORARIOS

jw.WritePropertyName("Horarios");

jw.WriteStartArray();

int k;

k = 0;

for (k = 0; k < this.Horarios.Count; k++)

{

jw.WriteStartObject();

jw.WritePropertyName("Dia");

jw.WriteValue(Horarios[k].DiaAtencionStr);

jw.WritePropertyName("HoraDesde");

jw.WriteValue(Horarios[k].HoraDesde);

jw.WritePropertyName("HoraHasta");

jw.WriteValue(Horarios[k].HoraHasta);

jw.WriteEndObject();

}

jw.WriteEndArray();

//PRECIOS

jw.WritePropertyName("Precios");

jw.WriteStartArray();

int l;

l = 0;

for (l = 0; l < this.Precios.Count; l++)

{

jw.WriteStartObject();

jw.WritePropertyName("TipoVehiculo");

jw.WriteValue(Precios[l].TipoVehiculoStr);

jw.WritePropertyName("Dia");

jw.WriteValue(Precios[l].DiaAtencionStr);

jw.WritePropertyName("Tiempo");

jw.WriteValue(Precios[l].TiempoStr);

jw.WritePropertyName("Monto");

jw.WriteValue(Precios[l].Monto);

jw.WriteEndObject();

}

jw.WriteEndArray();

jw.WriteEndObject();

return sb.ToString();

}

# JSON y JavaScript

En esta sección se explicara como es el manejo de JSON con JavaScript y su interacción, dando métodos y ejemplo utilizados.

* En el ejemplo siguiente se usa **JSON.parse** para convertir una cadena JSON en un objeto.

var jsontext = {"firstname":"Jesper","surname":"Aaberg","phone":["555-0100","555-0120"]}';

var contact = JSON.parse(jsontext);

document.write(contact.surname + ", " + contact.firstname);

* En el ejemplo siguiente se usa **JSON.stringify** para convertir un objeto en una cadena JSON

var nuevaCadena = JSON.stringify(libros);

alert(nuevaCadena);

**Se vera**

'[{"Titulo": "El señor de los anillos", "Autor": "J.R.R. Tolkien"},

{"Titulo": "Cancion de hielo y fuego", "Autor": "George RR Martin"},

{"Titulo": "Los Pilares de la Tierra", "Autor": "Ken Follett"}]';

# Referencias

* <http://json.org/>
* <http://www.json.org/js.html>
* <http://jhernandz.es/noticia/uso-json-con-javascript>
* <http://www.devjoker.com/contenidos/catss/459/Serializacion-JSON-con-NET.aspx>
* <http://james.newtonking.com/json>
* <http://www.devjoker.com/contenidos/catss/505/JSONNET-Mejorando-la-serializacion-JSON-con-NET.aspx>
* <http://stackoverflow.com/questions/6201529/turn-c-sharp-object-into-a-json-string-in-net-4>
* <http://www.brian-driscoll.com/2012/11/just-code-tojson-and-toxml-object.html?goback=.gde_136510_member_185144755>
* <http://speakingin.net/2007/10/04/trucos-creando-un-metodo-de-extension-tojson-con-net-35/>