Settings & Attributes

Xavier Morera
HELPING DEVELOPERS UNDERSTAND SEARCH & BIG DATA

@xmorera www.xaviermorera.com

Settings & Attributes

User preference

Provided during conversion

Customize serialization process

Settings

Declarative tag

Specified on classes, properties...

Control serialization & deserialization

Attributes

Settings

DateFormatHandling*

MissingMemberHandling

ReferenceLoopHandling

NullValueHandling

DefaultValueHandling

ObjectCreationHandling

TypeNameHandling

Binder

ConstructorHandling

Converters

ContractResolver

TraceWriter

Error

Attributes

Json.NET	.NET Attributes
JsonObjectAttribute	SerializableAttribute
JsonArrayAttribute	DataContractAttribute
JsonDictionaryAttribute	DataMemberAttribute
JsonPropertyAttribute	NonSerializedAttribute
JsonIgnoreAttribute	
JsonConverterAttribute	
JsonExtensionDataAttribute	
JsonConstructorAttribute	

```
JsonSerializerSettings jsonSettingsIgnore = new JsonSerializerSettings
{
    MissingMemberHandling = MissingMemberHandling.Ignore
};
```

Handling Missing Members

MissingMemberHandling specifies behavior when a JSON text contains a member

- With no equivalent property in the class that it is being deserialized to

Options are

- Ignore or Error

Demo

Settings: MissingMemberHandling

```
JsonSerializerSettings settingsAll = new JsonSerializerSettings
{
    ReferenceLoopHandling = ReferenceLoopHandling.Ignore
};
```

Circular References

ReferenceLoopHandling setting specifies how to handle reference loops

- Loops can be ignored, an error is raised, or serialize references

Use PreserveReferencesHandling to reference objects

Demo

Managing Circular References with the ReferenceLoopHandling Setting

```
JsonSerializerSettings jsonSettingsIgnore = new JsonSerializerSettings
{
    NullValueHandling = NullValueHandling.Ignore
};
```

Handling Null Values

Specify how to handle when a null value is encountered with NullValueHandling

Property can be serialized with a null value

- Or property is ignored and not included in the resulting JSON

Demo

Handling null Values with NullValueHandling

```
[DefaultValue("Passionate about teaching")]
public string state { get; set; }
```

Default Values

Attribute used to specify how to handle default values

- For serialization and deserialization

Can be included or ignored

Demo

Default Value Attribute and DefaultValueHandling Setting

```
new JsonSerializerSettings
{
    ObjectCreationHandling = ObjectCreationHandling.Replace
});
```

Object Creation

Control behavior on object creation

Via the ObjectCreationHandling setting

Append or replace values

Demo

Controlling object creation with the ObjectCreationHandling setting

```
JsonSerializerSettings settingsAll = new JsonSerializerSettings
{
    TypeNameHandling = TypeNameHandling.All
};
```

Type Name Handling

Type name information usually not preserved on serialization

- May be important for deserialization

Use the TypeNameHandling setting

Demo

Preserving Type Information with TypeNameHandling

```
new JsonSerializerSettings
{
    TypeNameHandling = TypeNameHandling.All,
    TypeNameAssemblyFormatHandling = TypeNameAssemblyFormatHandling.Full
});
```

Assembly Name

The assembly name is preserved with TypeNameHandling

- Potentially, multiple assembly versions

Preserve the full assembly name and version with TypeNameAssemblyFormatHandling

- Replaces TypeNameAssemblyFormat (obsolete)

Demo

Full Assembly Name with TypeNameAssemblyFormatHandling

```
JsonSerializerSettings settings = new JsonSerializerSettings
{
    TypeNameHandling = TypeNameHandling.All,
    SerializationBinder = binder
};
```

Custom Serialization Binder

Get type and assembly information

- Use TypeNameHandling and TypeNameAssemblyFormatHandling

Create a custom serializer to control serialization output

Demo

Custom Serializer with TypeSerializationBinder

```
new JsonSerializerSettings
{
    TypeNameHandling = TypeNameHandling.All,
    MetadataPropertyHandling = MetadataPropertyHandling.ReadAhead
});
```

Metadata Handling

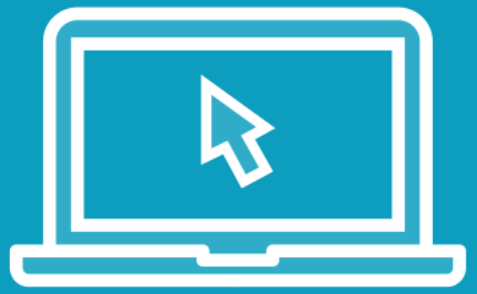
Json.NET focuses on speed

- Reads and processes properties in order

Can configure to read all properties with MetadataPropertyHandling

- Affects performance

Demo



Settings: MetadataPropertyHandling

```
new JsonSerializerSettings
{
    ConstructorHandling = ConstructorHandling.AllowNonPublicDefaultConstructor
};
```

Constructor Handling

Can specify which constructor to use with ConstructorHandling

- **Even a non-public constructor**

Demo

Settings: ConstructorHandling

Attributes

Json.NET

- JsonObjectAttribute
- JsonArrayAttribute
- JsonDictionaryAttribute
- JsonPropertyAttribute
- JsonIgnoreAttribute
- JsonConverterAttribute
- JsonExtensionDataAttribute
- JsonConstructorAttribute

Standard .NET Attributes

- SerializableAttribute
- DataContractAttribute
- DataMemberAttribute
- NonSerializedAttribute

```
[JsonObject()]
public class AuthorJsonObject : Author
{
    public AuthorJsonObject()
    {
        courses = new List<Course>();
    }
}
```

Attributes

Control the serialization process

Directly from the class

Demo

Attributes

```
[JsonObject(MemberSerialization = MemberSerialization.OptOut)]
public class AuthorJsonObjectOptOut
{
        ...
}
```

MemberSerialization Enumeration

Control which members are included or excluded

- Use MemberSerialization enumeration

Specify OptIn, OptOut, or Fields

Demo

MemberSerialization OptIn and OptOut

```
[JsonProperty(PropertyName = "AuthorName", Required = Required.Always)]
public string name { get; set; }
```

JsonPropertyAttribute

Can be used to specify if a field is required

- Among other useful properties

Demo

JsonProperty

```
[JsonConverter(typeof(StringEnumConverter))]
public Relationship relationship { get; set; }

[JsonConverter(typeof(JavaScriptDateTimeConverter))]
public DateTime since { get; set; }
```

JsonConverter Attribute

Specify JsonConverter using an attribute

- Provides greater control of the serialization process

Demo

JsonConverter Attribute

[JsonConstructor]

AuthorConstructorAttribute(string authorName)

JsonConstructor Attribute

Specify constructor to use on deserialization

- Using the JsonConstructor attribute

Demo

JsonConstructor Attribute

```
[JsonExtensionData]
public IDictionary<string, JToken> unMappedFields;
```

JsonExtensionData Attribute

Json.NET can be configured to respond to multiple scenarios

- For example missing members
  - Raise an exception or ignore

Use JsonExtensionData to store all unmapped fields in a collection

Demo

JsonExtensionData Attribute

Takeaway

Serialization and deserialization

- Main functionalities of Json.NET
- Json ←→.NET objects

JsonConvert provides a quick conversion

- May want to customize the process
- Use settings and attributes

Settings

- Control on conversion

Attributes

- Placed on classes and members