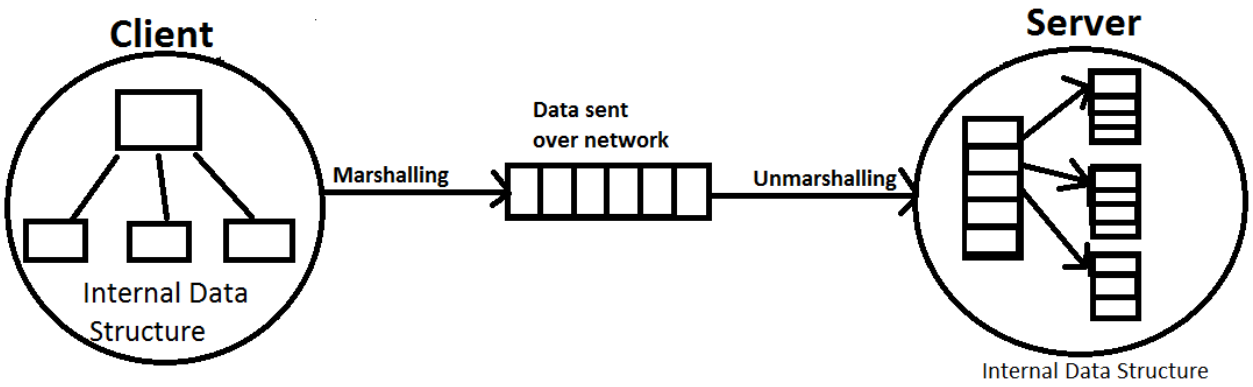
MISA®
TIN CẬY - TIỆN ÍCH - TẬN TÌNH

# Serialization và Model Binding

Nguyễn Văn Mạnh

# A. Serialization

- Định nghĩa: nói về mã hóa cấu trúc dữ liệu.



# A. Serialization

- Nội dung:
- 1. Media Formatter trong Web API 2
- 2. JSON và XML Serialization
- 3. BSON
- 4. Content Negotiation
- 5. Model Validation
- 6. Parameter Binding

# 1. Media Formatter trong Web API 2

- Internet Media Type
  - Media Type (MIME type)
    - text/html
    - image/png
    - application/json
  - Content-Type header của HTTP message

```
HTTP/1.1 200 OK
Content-Length: 95267
Content-Type: image/png
```

  - Accept header

```
Accept: text/html,application/xhtml+xml,application/xml
```

# 1. Media Formatter trong Web API 2

- Media Type quyết định cách mà Web API serialize và deserialize thân HTTP message
  - Web API hỗ trợ XML, JSON, BSON và form-urlencoded

JSON BSON

XML

# 1. Media Formatter trong Web API 2

- Tự tạo media formatter từ một trong các class:
  - MediaTypeFormatter: class sử dụng đọc ghi bất đồng bộ
  - BufferedMediaTypeFormatter: thừa kế từ MediaTypeFormatter nhưng sử dụng phương thức đọc ghi đồng bộ => Đơn giản hơn
  - Ví dụ:

```
namespace MISA.ProductStore.Formatters
{
    2 references
    public class ProductCsvFormater : BufferedMediaTypeFormatter
    {
        1 reference | 0 exceptions
        public ProductCsvFormater()
        {
            SupportedMediaTypes.Add(new System.Net.Http.Headers.MediaTypeHeaderValue("text/csv"));
        }
```

# 2. JSON và XML Serialization

- JSON Media-Type Formatter:
  - JSON format cung cấp bởi lớp JsonMediaTypeFormatter mặc định sử dụng thư viện Json.NET
  - Thay đổi mặc định

```
//JsonMediaTypeFormatter  sử dụng DataContractJsonSerializer thay cho Json.NET
var json = GlobalConfiguration.Configuration.Formatters.JsonFormatter;
json.UseDataContractJsonSerializer = true;
```

# 2. JSON và XML Serialization

- JSON Media-Type Formatter:

  1. JSON serialization

  - Mặc định properties, fields public được bao gồm trong serialization

  - [JsonIgnore]

```
public class Product
{
    public string Name { get; set; }
    public decimal Price { get; set; }
    [JsonIgnore]
    public int ProductCode { get; set; } // omitted
}
```

# 2. JSON và XML Serialization

- JSON Media-Type Formatter:

    1. JSON serialization

    - [DataContract]

    - [DataMember]

```
[DataContract]
public class Product
{
    [DataMember]
    public string Name { get; set; }
    [DataMember]
    public decimal Price { get; set; }
    public int ProductCode { get; set; }  // omitted by default
}
```

# 2. JSON và XML Serialization

- JSON Media-Type Formatter:

    2. Read-Only properties

        - Mặc định được serializer

    3. Date

        - Mặc định: theo chuẩn định dạng ISO 8601

        - UTC/Local

```
2012-07-27T18:51:45.53403Z          // UTC
2012-07-27T11:51:45.53403-07:00     // Local
```

```
// Convert all dates to UTC
var json = GlobalConfiguration.Configuration.Formatters.JsonFormatter;
json.SerializerSettings.DateTimeZoneHandling = Newtonsoft.Json.DateTimeZoneHandling.Utc;
```

# 2. JSON và XML Serialization

- JSON Media-Type Formatter:

    4. Indenting

```
var json = GlobalConfiguration.Configuration.Formatters.JsonFormatter;
json.SerializerSettings.Formatting = Newtonsoft.Json.Formatting.Indented;
```

    5. Camel Casing

```
var json = GlobalConfiguration.Configuration.Formatters.JsonFormatter;
json.SerializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver();
```

# 2. JSON và XML Serialization

- JSON Media-Type Formatter:

## 6. Anonymous Object

```
public object Get()
{
    return new {
        Name = "Alice",
        Age = 23,
        Pets = new List<string> { "Fido", "Polly", "Spot" }
    };
}
```

Response body

```
{"Name":"Alice","Age":23,"Pets":["Fido","Polly","Spot"]}
```

# 2. JSON và XML Serialization

- JSON Media-Type Formatter:

  7. Nhận JSON object cấu trúc lỏng lẻo:

  - Deserialize request body thành kiểu Newtonsoft.Json.Linq.JObject

```
public void Post(JObject person)
{
    string name = person["Name"].ToString();
    int age = person["Age"].ToObject<int>();
}
```

- Chú ý: XML serializer không hỗ trợ Anonymous và JObject

# 2. JSON và XML Serialization

- XML Media-Type Formatter:
  - XML format cung cấp bởi lớp XmlMediaTypeFormatter mặc định sử dụng class DataContractSerializer
  - Thay đổi mặc định

```
//XmlMediaTypeFormatter sử dụng XmlSerializer thay cho DataContractSerializer
var xml = GlobalConfiguration.Configuration.Formatters.XmlFormatter;
xml.UseXmlSerializer = true;
```

# 2. JSON và XML Serialization

- XML Media-Type Formatter:
  - 1. XML serialization mặc định

- Public properties, fields (read/write) => Serialized

- [IgnoreDataMember] => loại bỏ khỏi Serialized

- Private, protected properties, fields => not Serialized

- Read-only properties => not Serialized

- Tên class, prop, fields trong XML chính xác như trong khai báo class

- [DataContract], [DataMember] (private, protected member)

# 2. JSON và XML Serialization

- XML Media-Type Formatter:

    2. Read-only properties

```
[DataContract]
public class Product
{
    [DataMember]
    private int pcode;  // serialized

    // Not serialized (read-only)
    public int ProductCode { get { return pcode; } }
}
```

    3. Date

    - Theo định dạng ISO 8601

# 2. JSON và XML Serialization

- XML Media-Type Formatter:

    4. Cài Per-Type XML Serializer

```
var xml = GlobalConfiguration.Configuration.Formatters.XmlFormatter;
// Use XmlSerializer for instances of type "Product":
xml.SetSerializer<Product>(new XmlSerializer(typeof(Product)));
```

# 2. JSON và XML Serialization

- Loại bỏ JSON hoặc XML Formatter
  - Lí do:

  - Giới hạn API response về media type cụ thể

  - Thay thế formatter mặc định bằng custom formatter

```
void ConfigureApi(HttpConfiguration config)
{
    // Remove the JSON formatter
    config.Formatters.Remove(config.Formatters.JsonFormatter);

    // or

    // Remove the XML formatter
    config.Formatters.Remove(config.Formatters.XmlFormatter);
}
```

# 2. JSON và XML Serialization

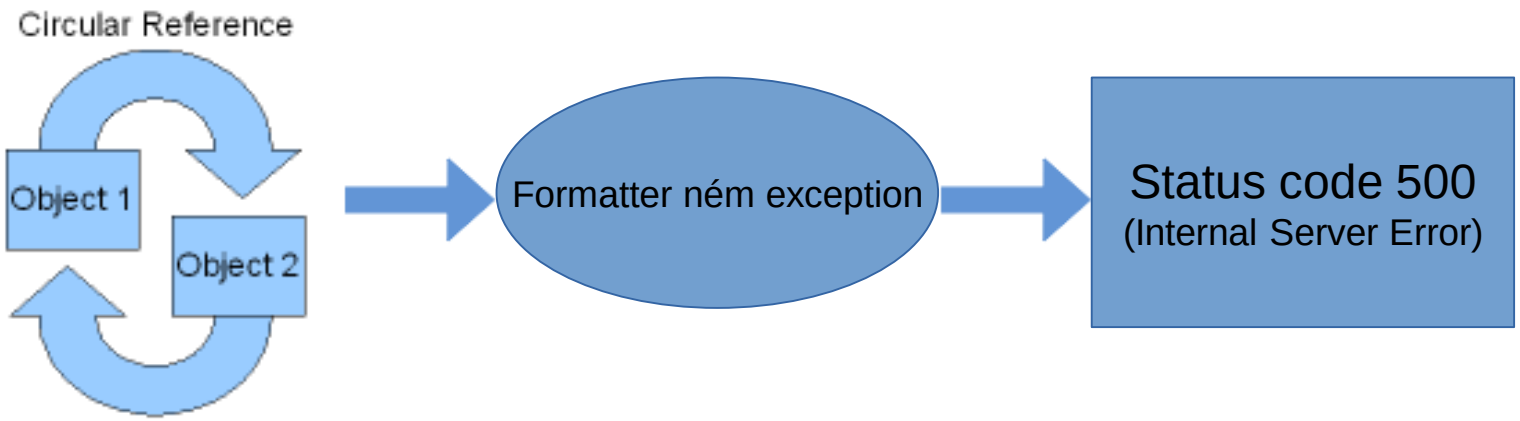
- Xử lí những quan hệ xoay vòng của Object
  - Vấn đề:

```
public class Employee
{
    public string Name { get; set; }
    public Department Department { get; set; }
}

public class Department
{
    public string Name { get; set; }
    public Employee Manager { get; set; }
}

public class DepartmentsController : ApiController
{
    public Department Get(int id)
    {
        Department sales = new Department() { Name = "Sales" };
        Employee alice = new Employee() { Name = "Alice", Department = sales };
        sales.Manager = alice;
        return sales;
    }
}
```

```
{
    "Name": "Sales",
    "Manager": {
        "Name": "Alice",
        "Department": {
        ....
        }
    }
}
```

# 2. JSON và XML Serialization

- Xử lí những quan hệ xoay vòng của Object
  - Vấn đề:



# 2. JSON và XML Serialization

- Xử lí những quan hệ xoay vòng của Object
  - Xử lí vấn đề trong JSON:

```
var json = GlobalConfiguration.Configuration.Formatters.JsonFormatter;
json.SerializerSettings.PreserveReferencesHandling =
    Newtonsoft.Json.PreserveReferencesHandling.All;
```

```
{"$id":"1","Name":"Sales","Manager":{"$id":"2","Name":"Alice","Department":{"$ref":"1"}}}
```

# 2. JSON và XML Serialization

- Xử lí những quan hệ xoay vòng của Object
  - Xử lí vấn đề trong XML:

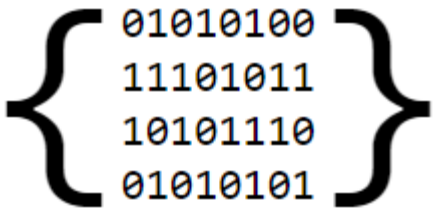
- [DataContract(IsReference=true)]

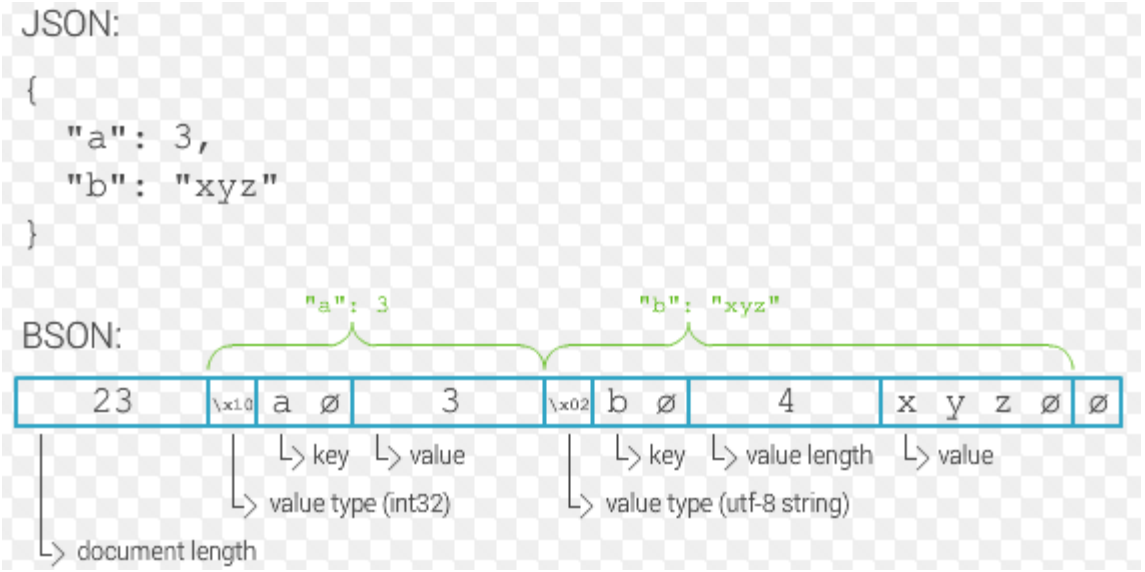
```
[DataContract(IsReference=true)]
public class Department
{
    [DataMember]
    public string Name { get; set; }
    [DataMember]
    public Employee Manager { get; set;
}
```

```
<Department xmlns:i="http://www.w3.org/2001/XMLSchema-instance" z:Id="i1"
            xmlns:z="http://schemas.microsoft.com/2003/10/Serialization/"
            xmlns="http://schemas.datacontract.org/2004/07/Models">
  <Manager>
    <Department z:Ref="i1" />
    <Name>Alice</Name>
  </Manager>
  <Name>Sales</Name>
</Department>
```

# 3. BSON

- BSON: Binary JSON, serialization định dạng nhị phân, dữ liệu kiểu số lưu dạng byte.



# 3. BSON

- Enable BSON trên server

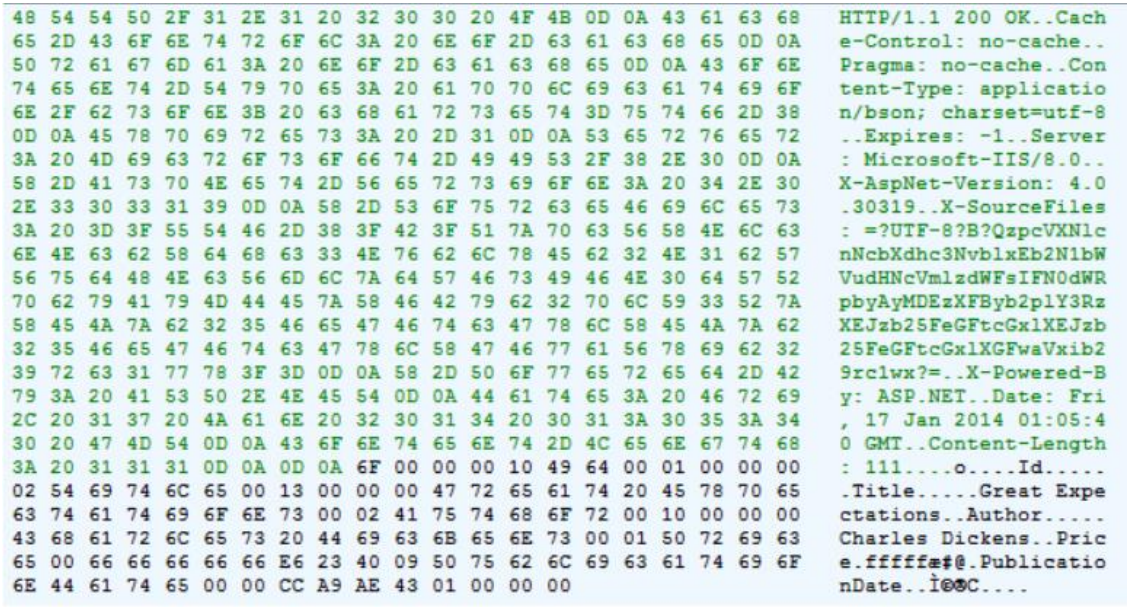
  - Client request “application/bson”, Web API sẽ sử dụng BSON Formater

```
public static class WebApiConfig
{
    public static void Register(HttpConfiguration config)
    {
        config.Formatters.Add(new BsonMediaTypeFormatter());

        // Other Web API configuration not shown...
    }
}
```

# 3. BSON

- Enable BSON trên server

  - Ví dụ:



```
48 54 54 50 2F 31 2E 31 20 32 30 30 20 4F 4B 0D 0A 43 61 63 68   HTTP/1.1 200 OK..Cach
65 2D 43 6F 6E 74 72 6F 6C 3A 20 6E 6F 2D 63 61 63 68 65 0D 0A   e-Control: no-cache..
50 72 61 67 6D 61 3A 20 6E 6F 2D 63 61 63 68 65 0D 0A 43 6F 6E   Pragma: no-cache..Con
74 65 6E 74 2D 54 79 70 65 3A 20 61 70 70 6C 69 63 61 74 69 6F   tent-Type: applicatio
6E 2F 62 73 6F 6E 3B 20 63 68 61 72 73 65 74 3D 75 74 66 2D 38   n/bson; charset=utf-8
0D 0A 45 78 70 69 72 65 73 3A 20 2D 31 0D 0A 53 65 72 76 65 72   ..Expires: -1..Server
3A 20 4D 69 63 72 6F 73 6F 66 74 2D 49 49 53 2F 38 2E 30 0D 0A   : Microsoft-IIS/8.0..
58 2D 41 73 70 4E 65 74 2D 56 65 72 73 69 6F 6E 3A 20 34 2E 30   X-AspNet-Version: 4.0
2E 33 30 33 31 39 0D 0A 58 2D 53 6F 75 72 63 65 46 69 6C 65 73   .30319..X-SourceFiles
3A 20 3D 3F 55 54 46 2D 38 3F 42 3F 51 7A 70 63 56 58 4E 6C 63   : =?UTF-8?B?QzpcVXNlc
6E 4E 63 62 58 64 68 63 33 4E 76 62 6C 78 45 62 32 4E 31 62 57   nNcbXdhc3NvblxEb2N1bW
56 75 64 48 4E 63 56 6D 6C 7A 64 57 46 73 49 46 4E 30 64 57 52   VudHNcVmlzdWFsIFN0dWR
70 62 79 41 79 4D 44 45 7A 58 46 42 79 62 32 70 6C 59 33 52 7A   pbyAyMDEzXFByb2plY3Rz
58 45 4A 7A 62 32 35 46 65 47 46 74 63 47 78 6C 58 45 4A 7A 62   XEJzb25FeGFtcGxlXEJzb
32 35 46 65 47 46 74 63 47 78 6C 58 47 46 77 61 56 78 69 62 32   25FeGFtcGxlXGFwaVxib2
39 72 63 31 77 78 3F 3D 0D 0A 58 2D 50 6F 77 65 72 65 64 2D 42   9rc1wx?=..X-Powered-B
79 3A 20 41 53 50 2E 4E 45 54 0D 0A 44 61 74 65 3A 20 46 72 69   y: ASP.NET..Date: Fri
2C 20 31 37 20 4A 61 6E 20 32 30 31 34 20 30 31 3A 30 35 3A 34   , 17 Jan 2014 01:05:4
30 20 47 4D 54 0D 0A 43 6F 6E 74 65 6E 74 2D 4C 65 6E 67 74 68   0 GMT..Content-Length
3A 20 31 31 31 0D 0A 0D 0A 6F 00 00 00 10 49 64 00 01 00 00 00   : 111....o....Id.....
02 54 69 74 6C 65 00 13 00 00 00 47 72 65 61 74 20 45 78 70 65   .Title.....Great Expe
63 74 61 74 69 6F 6E 73 00 02 41 75 74 68 6F 72 00 10 00 00 00   ctations..Author.....
43 68 61 72 6C 65 73 20 44 69 63 6B 65 6E 73 00 01 50 72 69 63   Charles Dickens..Pric
65 00 66 66 66 66 66 E6 23 40 09 50 75 62 6C 69 63 61 74 69 6F   e.fffffæ#@.Publicatio
6E 44 61 74 65 00 00 CC A9 AE 43 01 00 00 00                     nDate..Ì©®C....
```

# 4. Content Negotiation

- Serialization

  - Ví dụ:

    - Client request

```
GET http://localhost.:21069/api/products/1 HTTP/1.1
Host: localhost.:21069
Accept: application/json, text/javascript, */*; q=0.01
```

    - Server response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 57
Connection: Close

{"Id":1,"Name":"Gizmo","Category":"Widgets","Price":1.99}
```

# 5. Model Validation

- Data Annotation
  - - Sử dụng attribute để cài Validation cho properties

```
using System.ComponentModel.DataAnnotations;

namespace MyApi.Models
{
    public class Product
    {
        public int Id { get; set; }
        [Required]
        public string Name { get; set; }
        public decimal Price { get; set; }
        [Range(0, 999)]
        public double Weight { get; set; }
    }
}
```

```
{ "Id":4, "Price":2.99, "Weight":5 }
```

# 5. Model Validation

- Xử lí lỗi Validation

  - Web API không tự động trả về lỗi cho client khi validation fails.

  => Xử lí trong action

# 5. Model Validation

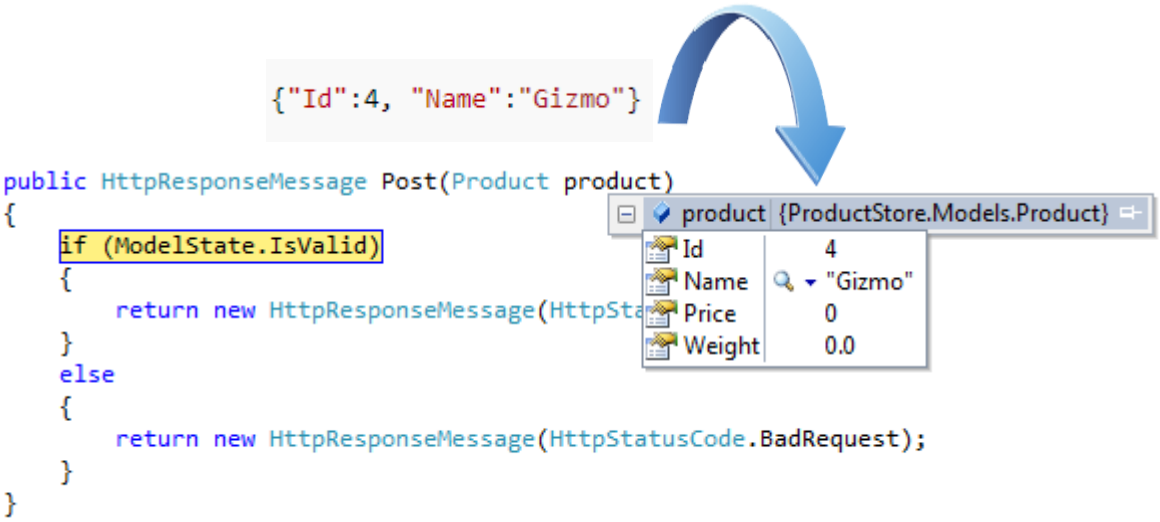
- Data Annotation

  - Sử dụng attributem để cài Validation cho properties

```
public class ProductsController : ApiController
{
    public HttpResponseMessage Post(Product product)
    {
        if (ModelState.IsValid)
        {
            // Do something with the product (not shown).

            return new HttpResponseMessage(HttpStatusCode.OK);
        }
        else
        {
            return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ModelState);
        }
    }
}
```

# 5. Model Validation

- Data Annotation



# 5. Model Validation

- Data Annotation

```
{"Id":4, "Name":"Gizmo", "Color":"Blue"}
```

JSON Formatter bỏ qua giá trị đó

# 5. Model Validation

- Data Annotation

  - Vấn đề khi client post dư thừa

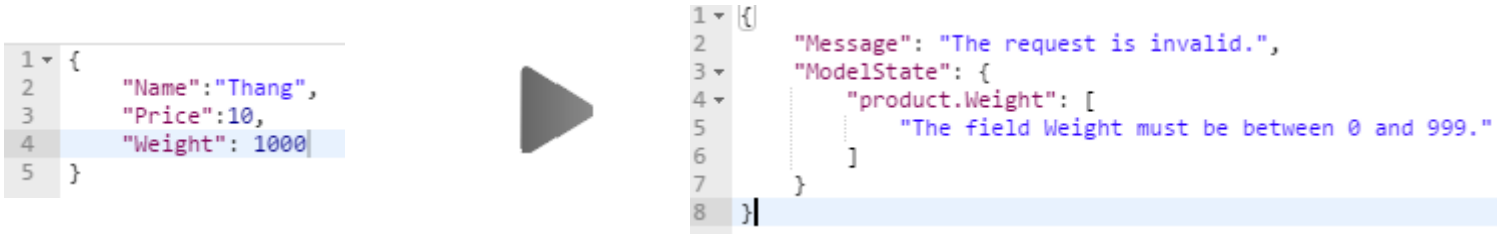
```
public class UserProfile
{
    public string Name { get; set; }
    public Uri Blog { get; set; }
    public bool IsAdmin { get; set; }  // uh-oh!
}
```

```
public class UserProfileDTO
{
    public string Name { get; set; }
    public Uri Blog { get; set; }
    // Leave out "IsAdmin"
}
```

# 5. Model Validation

- Xử lí lỗi Validation

  - Ví dụ:



# 5. Model Validation

- Xử lí lỗi Validation

  - Tạo một  action filter kiểm tra model state trước khi action được gọi

```
public class ValidateModelAttribute : ActionFilterAttribute
{
    public override void OnActionExecuting(HttpActionContext actionContext)
    {
        if (actionContext.ModelState.IsValid == false)
        {
            actionContext.Response = actionContext.Request.CreateErrorResponse(
                HttpStatusCode.BadRequest, actionContext.ModelState);
        }
    }
}
```

# 5. Model Validation

- Xử lí lỗi Validation

  - Sau khi tạo action filter:

    1. Filter mọi controller

```
public static class WebApiConfig
{
    public static void Register(HttpConfiguration config)
    {
        config.Filters.Add(new ValidateModelAttribute());

        // ...
    }
}
```

    2. Cho controller hoặc action riêng lẻ

```
[ValidateModel]
public HttpResponseMessage Post(Product product)
```

# 6. Parameter Binding

- Type Converter

```
class GeoPointConverter : TypeConverter
```

```
[TypeConverter(typeof(GeoPointConverter))]
public class GeoPoint
```


THANK YOU