Tipo : Guía de Enunciado
Capítulo : ASP NET Web Api
Duración : 60 minutos

I. OBJETIVO

Optimizando Web Api con compresión de respuestas.

II. REQUISITOS

Los siguientes elementos de software son necesarios para la realización del laboratorio:

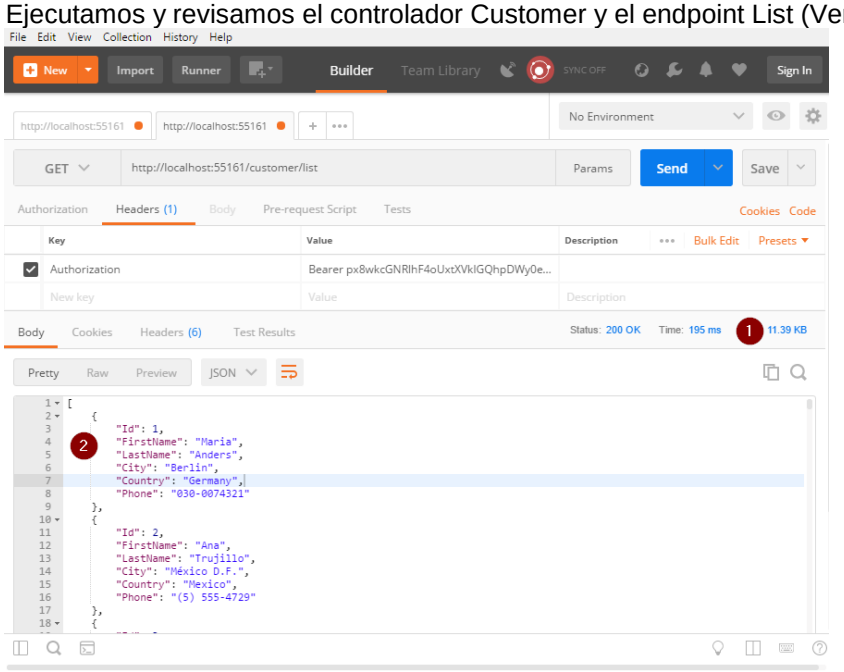
- Windows 10 (como mínimo Windows 8)
- Visual Studio 2017 (como mínimo Visual Studio 2015)

III. EJECUCIÓN DEL LABORATORIO

1. Abrir la solución del módulo anterior.

2. Ubicarse en el proyecto "**Cibertec.WebApi**"

3. Ejecutamos y revisamos el controlador Customer y el endpoint List (Ver imagen)



En la imagen notaremos:

(1) El tamaño del response es de 11.39kb (**Puede variar según entorno**), lo cual para una respuesta web es un poco elevada
(2) La notación de los objetos resultantes no son CamelCase, un estándar para el trabajo con JSON.

Si revisamos un poco más notaremos que dichos responses no implementan compresión "**gzip**"

Uno de los objetivos primordiales de implementar una WebApi es que el response sea lo más liviano posible y que aplique un formato estándar para poder usado con los diferentes frameworks javascript que existen.

4. Procedemos a instalar el siguiente paquete:

   - Install-Package Microsoft.AspNet.WebApi.Extensions.Compression.Server.Owin

5. En la carpeta "**App_Start**" creamos la clase "**WebApi.config**" con el siguiente código:

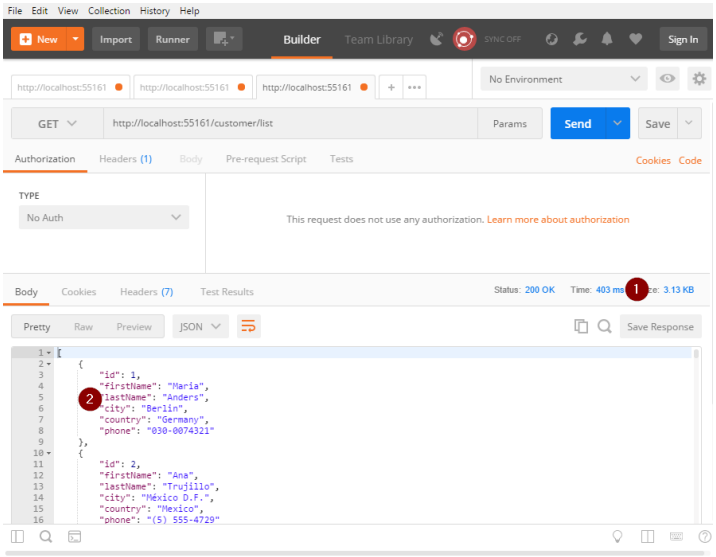
```
using Microsoft.AspNet.WebApi.Extensions.Compression.Server;
using Newtonsoft.Json.Serialization;
using System.Net.Http.Extensions.Compression.Core.Compressors;
using System.Web.Http;

namespace Cibertec.WebApi
{
    public static class WebApiConfig
    {
        public static void Configure(HttpConfiguration config)
        {
            config.MessageHandlers.Insert(0, new
ServerCompressionHandler(new GZipCompressor(), new DeflateCompressor()));

config.Formatters.JsonFormatter.SerializerSettings.ContractResolver = new
CamelCasePropertyNamesContractResolver();
        }
    }
}
```

6. Una vez configurado procedemos a editar nuestra clase "**Startup**" de la siguiente manera:

```
using Cibertec.WebApi.Handlers;
using Microsoft.Owin;
using Owin;
using System.Web.Http;
using System.Web.Http.ExceptionHandling;

[assembly: OwinStartup(typeof(Cibertec.WebApi.Startup))]

namespace Cibertec.WebApi
{
    public class Startup
    {
        public void Configuration(IAppBuilder app)
        {
```

```
            log4net.Config.XmlConfigurator.Configure();
            var log = log4net.LogManager.GetLogger(typeof(Startup));
            log.Debug("Logging is enabled");

            var config = new HttpConfiguration();
            config.Services.Replace(typeof(IExceptionHandler), new
GlobalExceptionHandler());

            DIConfig.ConfigureInjector(config);
            TokenConfig.ConfigureOAuth(app, config);
            RouteConfig.Register(config);
            WebApiConfig.Configure(config);
            app.UseWebApi(config);
        }
    }
}
```

7. Procedemos a ejecutar la aplicación y validamos el resultado.



(1) Ahora el resultado es 3.13kb, eso significa una reducción del 73% en el transporte del response.
(2) La notación concuerda con el estándar.

Validamos el header:

Como podemos notar la compresión gzip esta realizando de manera correcta, ayudándonos a reducir el tamaño del response a procesar.

IV. EVALUACIÓN

1. ¿Por qué para vistas parciales se deben de usar "_" el guion bajo?

   Bajo el lenguaje Razor, el guion bajo permite evitar mostrar Web Pages cuando son solicitadas directamente como parte de un request.