

CLASE 5

Laboratorio

El objetivo de este laboratorio es aplicar a un proyecto webapi los siguientes filtros básicos:

- **Filtros de autorización** que determinan si un usuario puede consumir una acción específica.
- **Filtros de excepción** que se ejecutan cuando hay una excepción no atrapada por un **try-catch** durante la ejecución de una acción, durante la ejecución de un filtro de acción, durante la creación de un controlador y durante el binding del modelo.

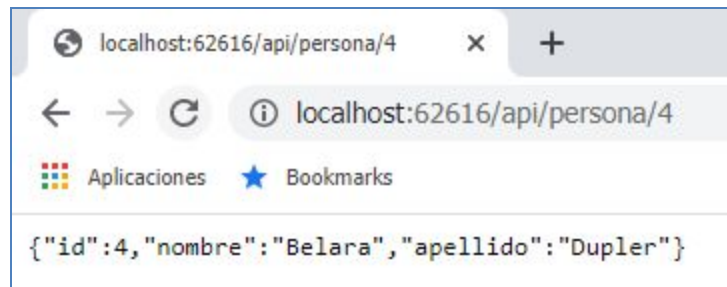
Nota: El proyecto base y punto de partida de esta ejercitación, se encuentra en la sección de descargas en el archivo **WebApiMiddleware_Inicial.zip**. Bájelo y luego de descomprimir el archivo, cargue la solución **WebApiMiddleware.sln**.

Para comprobar el correcto funcionamiento del proyecto y antes de ejercitar, ejecute la aplicación y pruebe las siguientes URLs, debería obtener estos resultados:

URL: <http://localhost:62616/api/persona>



URL: <http://localhost:62616/api/persona/4>



Pasos de la ejercitación:

1. Agregaremos un **filtro de autorización**, indicando que una acción solamente puede ser accedida por determinados usuarios que cumplan con alguna condición.

En la clase **startup** iniciamos el servicio de autenticación lo siguiente en el método **ConfigureServices**:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme).AddJwtBearer();
}
```

```

        services.AddMvc().AddXmlSerializerFormatters();
    }

```

2. Recuerde agregar el namespace para usar el tipo de autenticación **JSON Web Token**:

```
using Microsoft.AspNetCore.Authentication.JwtBearer;
```

3. Luego en el método **Configure** active la autenticación antes del middleware MVC dado que esto es previo a la carga del servicio MVC.

```

public void Configure(IApplicationBuilder app)
{
    app.UseAuthentication();
    app.UseMvc();
}

```

4. En el **PersonaController** y a nivel de acción agregamos el atributo **[Authorize]** con el namespace correspondiente

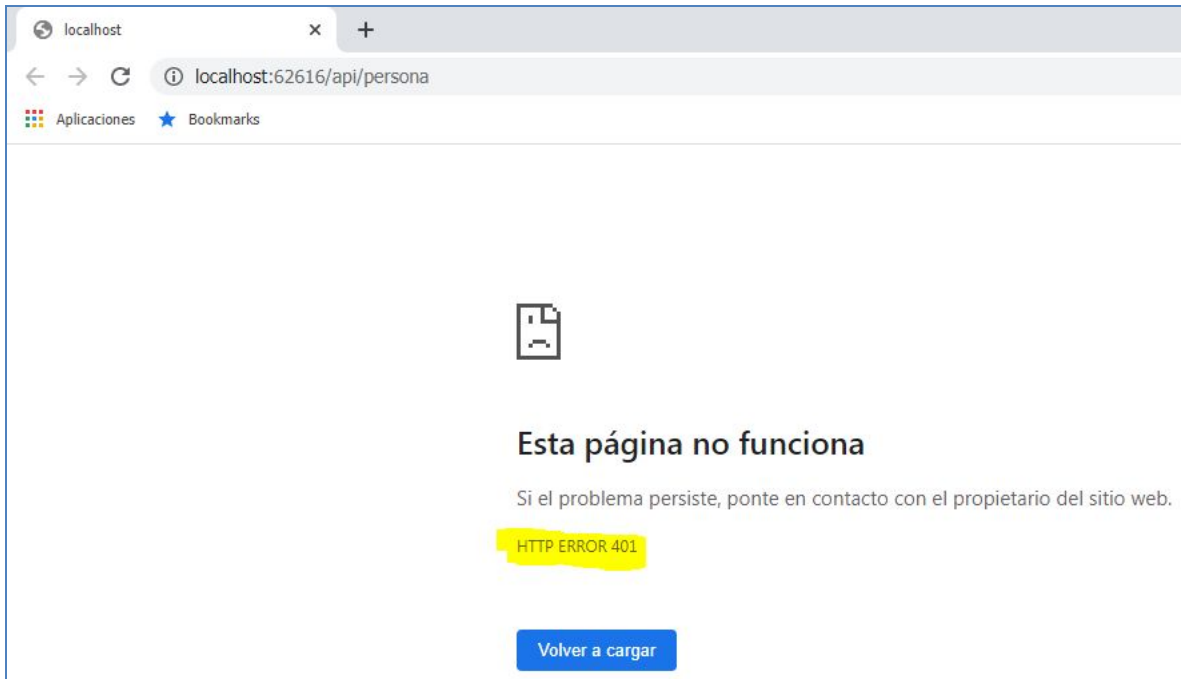
```
using Microsoft.AspNetCore.Authorization;
```

```

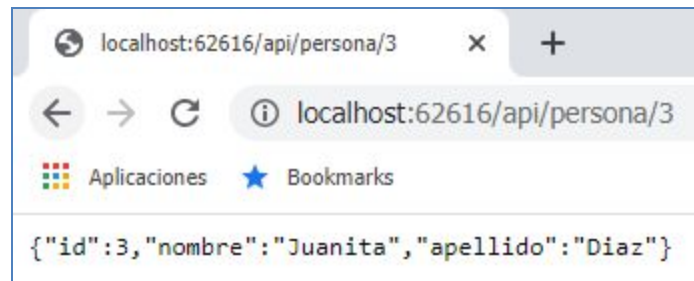
[HttpGet]
[Produces("application/xml")]
[Authorize]
public List<Persona> ListarTodas()
{
    return _listaPersonas;
}

```

5. Ejecute el webapi y trate de acceder a la acción que recupera la lista de personas, obtendrá el error **HTTP Error 401** que significa **unauthorized**.



6. Si prueba la otra acción que no tiene el filtro de autorización podrá acceder para ello ingrese la URL <http://localhost:62616/api/persona/3>



7. Recuerde que también podemos utilizar el filtro de autorización a nivel del controlador. Pruebe ubicarlo a nivel de api, ejecute y verá que no puede acceder al webapi obteniendo el error HTTP ERROR 401:

```
namespace WebApiMiddleware.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    [Authorize]

    public class PersonaController : ControllerBase
```

8. Probaremos ahora un **filtro global de excepción**. Agregue al proyecto un filtro llamado **FiltroExcepcion** en una carpeta **Filtros** ubicada a nivel del proyecto. Este filtro heredará de **ExceptionHandler** y deberá sobrescribir el evento **OnException**. El código obtenido será el siguiente, observe que es un tratamiento global de distintos tipos de excepciones y según el tipo puede personalizar el mensaje de error y retornarlo en algún tipo de objeto, para el ejemplo usamos un objeto **HttpResponse**:

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc.Filters;
using System;
using System.Net;

namespace WebApiMiddleware.Filtros
{
    public class FiltroExcepcion:ExceptionHandlerAttribute
    {
        public override void OnException(ExceptionContext context)
        {
            //Control global de excepciones
            //Analiza la excepción y personaliza el mensaje de error.
            //Puede aquí también agregar excepciones personalizadas y no
            propias del framework
            HttpStatusCode status = HttpStatusCode.InternalServerError;
            String message = String.Empty;

            var exceptionType = context.Exception.GetType();
            if (exceptionType == typeof(UnauthorizedAccessException))
            {
                message = "Acceso no autorizado!";
                status = HttpStatusCode.Unauthorized;
            }
            else if (exceptionType == typeof(NotImplementedException))
            {
                message = "Atención: Error en el servidor!";
                status = HttpStatusCode.NotImplemented;
            }
            else if (exceptionType == typeof(NullReferenceException))
            {
                message = "Valor nulo o desconocido!";
                status = HttpStatusCode.InternalServerError;
            }
            else
            {
                message = context.Exception.Message;
                status = HttpStatusCode.NotFound;
            }
        }
    }
}
```

```

context.ExceptionHandled = true;

HttpResponse response = context.HttpContext.Response;
response.StatusCode = (int)status;
response.ContentType = "application/json";
var err = message + " " + context.Exception.StackTrace;
response.WriteAsync(err);
}
}

```

9. Ubicamos un break en el método **OnException**, y configuraremos el filtro de excepción en la clase **startup**, en el método **ConfigureServices** y dentro de **AddMvc** configuramos las siguientes opciones para el filtro global:

```

using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Builder;
using Microsoft.Extensions.DependencyInjection;
using WebApiMiddleware.Filtros;

namespace WebApiExample
{
    public class Startup
    {
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme).AddJwtBearer();

            services.AddMvc(options =>
            {
                options.Filters.Add(new FiltroExcepcion());
            }).AddXmlSerializerFormatters();
        }
    }
}

```

10. En el controlador comente la acción **ListarPersonas** y solo para probar la excepción global cámbiela por esta acción:

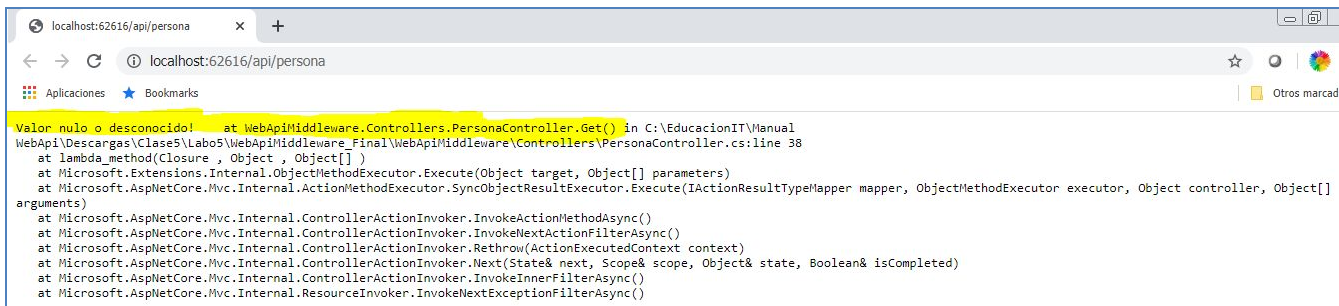
```

[HttpGet]
public IEnumerable<string> Get()
{
    string[] arrRetValues = null;
    if (arrRetValues.Length > 0)
    { }
    return arrRetValues;
}

```

11. Ejecute el webapi y observe que al llamar al Get, da la excepción **NullReferenceException** con el mensaje personalizado e información adicional sobre la excepción y dónde ocurrió.

```
Valor nulo o desconocido! at WebApiMiddleware.Controllers.PersonaController.Get() in
C:\EducacionIT\Manual
WebApi\Descargas\Clase5\Labo5\WebApiMiddleware_Final\WebApiMiddleware\Controllers\PersonaCont
roller.cs:line 38
    at lambda_method(Closure , Object , Object[] )
    at Microsoft.Extensions.Internal.ObjectMethodExecutor.Execute(Object target, Object[] parameters)
    at
Microsoft.AspNetCore.Mvc.Internal.ActionMethodExecutor.SyncObjectResultExecutor.Execute(IActionRe
sultTypeMapper mapper, ObjectMethodExecutor executor, Object controller, Object[] arguments)
    at Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker.InvokeActionMethodAsync()
    at Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker.InvokeNextActionFilterAsync()
    at Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker.Rethrow(ActionExecutedContext context)
    at Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker.Next(State& next, Scope& scope,
Object& state, Boolean& isCompleted)
    at Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker.InvokeInnerFilterAsync()
    at Microsoft.AspNetCore.Mvc.Internal.ResourceInvoker.InvokeNextExceptionFilterAsync()
```



The screenshot shows a web browser window with the address bar displaying 'localhost:62616/api/persona'. The browser's developer tools are open, showing a JavaScript error. The error message is 'Valor nulo o desconocido!' (Null or unknown value). The stack trace is as follows:

```
Valor nulo o desconocido! at WebApiMiddleware.Controllers.PersonaController.Get() in C:\EducacionIT\Manual
WebApi\Descargas\Clase5\Labo5\WebApiMiddleware_Final\WebApiMiddleware\Controllers\PersonaController.cs:line 38
    at lambda_method(Closure , Object , Object[] )
    at Microsoft.Extensions.Internal.ObjectMethodExecutor.Execute(Object target, Object[] parameters)
    at Microsoft.AspNetCore.Mvc.Internal.ActionMethodExecutor.SyncObjectResultExecutor.Execute(IActionResultTypeMapper mapper, ObjectMethodExecutor executor, Object controller, Object[]
arguments)
    at Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker.InvokeActionMethodAsync()
    at Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker.InvokeNextActionFilterAsync()
    at Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker.Rethrow(ActionExecutedContext context)
    at Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker.Next(State& next, Scope& scope, Object& state, Boolean& isCompleted)
    at Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker.InvokeInnerFilterAsync()
    at Microsoft.AspNetCore.Mvc.Internal.ResourceInvoker.InvokeNextExceptionFilterAsync()
```

Nota: El proyecto resultante luego de esta ejercitación, se encuentra en la sección de descargas en el archivo **WebApiMiddleware_Final.zip**.