Full Stack Snippets.

From Chris' Full Stack Blog.

PatchFiltererService

Filter out unwanted properties from your models on the server side in .NET.

From post: C# .NET Core and TypeScript: Using Generics and LINQ to Secure and Filter Operations on Your JSONPatchDocuments

PatchFiltererService.cs

```
using System;
using System.Linq;
using Microsoft.AspNetCore.JsonPatch;

namespace JsonPatchFilterExample.Services
{
    // a security filter for JSON patch filter operations
    // see the full blog post at https://chrisfrew.in/blog/filtering-json-patch-in-c-sharp/
    public static class PatchFiltererService
    {
        public static JsonPatchDocument<T> ApplyAttributeFilterToPatch<T, TU>(JsonPatchDocument<T> patch)
        where T : class
        where TU : Attribute
        {
            // Get path for all attributes of type TU that are in type T
            var allowedPaths = typeof(T)
```

```
                .GetProperties()
                .Where(x => x.GetCustomAttributes(false).OfType<TU>().Any())
                .Select(x => x.Name);

            // Now build a new JSONPatchDocument based on properties in T that
were found above
            var filteredPatch = new JsonPatchDocument<T>();
            patch.Operations.ForEach(x =>
            {
                if (allowedPaths.Contains(x.path))
                {
                    filteredPatch.Operations.Add(x);
                }
            });

            return filteredPatch;
        }
    }
}
```

Usage

```
using System;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.JsonPatch;
using JsonPatchFilterExample.Services;
using JsonPatchFilterExample.Models;
using System.ComponentModel.DataAnnotations;
using Microsoft.Extensions.FileProviders;
using System.IO;
using Newtonsoft.Json;

namespace JsonPatchFilterExample.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class WidgetController : ControllerBase
    {
        [HttpPatch("{id}")]
        public ActionResult Patch(Guid id, [FromBody]
JsonPatchDocument<WidgetModel> patch)
        {
            try
            {
                // For now, load the widget from the json file - ideally this
would be retrieved via a repository from a database
```

```
                var physicalProvider = new
PhysicalFileProvider(Directory.GetCurrentDirectory());
                var jsonFilePath = Path.Combine(physicalProvider.Root,
"App_Data", "ExampleWidget.json");
                var item = new WidgetModel();
                using (var reader = new StreamReader(jsonFilePath))
                {
                    var content = reader.ReadToEnd();
                    item = JsonConvert.DeserializeObject<WidgetModel>(content);
                }
                if (item.Id != id || patch == null)
                {
                    return NotFound();
                }

                // Create a new patch to match only the type and attributes
passed
                patch =
PatchFiltererService.ApplyAttributeFilterToPatch<WidgetModel,
StringLengthAttribute>(patch);

                // Apply the patch!
                patch.ApplyTo(item);

                // Update updated time - normally would be handled in a
repository
                item.Updated = DateTime.Now;

                // Update the item - ideally this would also be done with a
repository via an 'Update' method
                // write JSON directly to a file
                var json = JsonConvert.SerializeObject(item);

                //write string to file
                System.IO.File.WriteAllText(jsonFilePath, json);

                return Ok();
            }
            catch
            {
                return UnprocessableEntity();
            }
        }
    }
}
```