Health Actuator

Description

Objective: The objective is to work with the Health Actuator to add an endpoint at /actuator/health to retrieve health contributors registered in the application's service container.

This excerise to understand the steeltoe is distributed (Nuget), how the components are using the application.

The code template already added the Nuget packages.

The requirement is write the code for adding the actuator for the below items in **Program.cs**.

[Note: You should just implement the code in Program.cs; the other files are provided by your reference.]

Health Actuator - /actuator/health

Health Actuator

The health actuator adds a new endpoint at /actuator/health. This endpoint retrieves health contributors that have been registered in the application's service container with either Steeltoe's IHealthContributor or Microsoft's IHealthRegistration to aggregate the health checks of all the application's dependencies. This endpoint use the HTTP status code of 200 when everything is OK and 503 when it isn't. Details of the health checks can be included in JSON-formatted responses, too.

Sample output for Health actuator :

WeatherForcastController.cs

```
// This file for your reference, should not modify this file


  using Microsoft.AspNetCore.Mvc;
  using Microsoft.Extensions.Logging;
  using System;
  using System.Collections.Generic;
  using System.Linq;
  using System.Threading.Tasks;


  namespace Eureka.Register.Example.Controllers
  {
    [ApiController]
    [Route("[controller]")]
    public class WeatherForecastController : ControllerBase
    {
      private static readonly string[] Summaries = new[]
      {
        "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering",
"Scorching"
      };


      private readonly ILogger<WeatherForecastController> _logger;


      public WeatherForecastController(ILogger<WeatherForecastController> logger)
      {
```

```
        _logger = logger;
    }

    [HttpGet]
    public IEnumerable<WeatherForecast> Get()
    {
        _logger.LogInformation("Hi there");

        var rng = new Random();
        return Enumerable.Range(1, 5).Select(index => new WeatherForecast
        {
            Date = DateTime.Now.AddDays(index),
            TemperatureC = rng.Next(-20, 55),
            Summary = Summaries[rng.Next(Summaries.Length)]
        })
        .ToArray();
    }
  }
}
```

Program.cs

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Steeltoe.Management.Endpoint.Health;
using Steeltoe.Management.Endpoint;
namespace ActuatorExample
{
  public class Program
  {
    public static void Main(string[] args)
    {
      var host = CreateHostBuilder(args).Build();
      host.Run();
    }
    public static IHostBuilder CreateHostBuilder(string[] args) =>
      Host.CreateDefaultBuilder(args)
        .ConfigureWebHostDefaults(webBuilder =>
        {
          webBuilder.ConfigureServices(services =>
          {
            services.AddControllers();
            services.AddHealthActuator(); // Health Actuator
          });
          webBuilder.Configure(app =>
```

```
            {
                app.UseRouting();
                app.UseAuthorization();
                app.UseEndpoints(endpoints =>
                {
                    endpoints.MapControllers();
                    endpoints.MapAllActuators();
                });
            });
        });
    }
}
```

Startup.cs

```
// This file for your reference, should not modify this file

using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Steeltoe.Management.Tracing;
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Eureka.Register.Example
{
  public class Startup
  {
    public Startup(IConfiguration configuration)
    {
      Configuration = configuration;
    }

    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the container.
    public void ConfigureServices(IServiceCollection services)
    {
      services.AddControllers();

      //Steeltoe distributed tracing
      services.AddDistributedTracing();
    }
```

```
    // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
      if (env.IsDevelopment())
      {
        app.UseDeveloperExceptionPage();
      }

      app.UseHttpsRedirection();

      app.UseRouting();

      app.UseAuthorization();

      app.UseEndpoints(endpoints =>
      {
        endpoints.MapControllers();
      });
    }
  }
}
```

WeatherForcast.cs

```
// This file for your reference, should not modify this file


using System;


namespace Eureka.Register.Example
{
   public class WeatherForecast
   {
      public DateTime Date { get; set; }


      public int TemperatureC { get; set; }


      public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);


      public string Summary { get; set; }
   }
}
```

Info Actuator

Description

Objective: The objective is to work with the Info Actuator to add an endpoint at /actuator/health to gather various information.

This excerise to understand the steeltoe is distributed (Nuget), how the components are using the application.

The code template already added the Nuget packages.

The requirement is write the code for adding the actuator for the below items in **Program.cs**.

[Note: You should just implement the code in Program.cs; the other files are provided by your reference.]

Info Actuator - /actuator/info

Info Actuator

The info actuator adds a new endpoint at /actuator/info. This function gathers all kinds of information like versioning information, select package information, and anything you'd like to include with your own custom IInfoContributor. Everything is formatted as JSON and included in the response.

Sample output for Info actuator :

WeatherForcastController.cs

```
// This file for your reference, should not modify this file


using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;


namespace Eureka.Register.Example.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class WeatherForecastController : ControllerBase
    {
        private static readonly string[] Summaries = new[]
        {
            "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering",
"Scorching"
        };


        private readonly ILogger<WeatherForecastController> _logger;


        public WeatherForecastController(ILogger<WeatherForecastController> logger)
        {
```

```
        _logger = logger;
    }

    [HttpGet]
    public IEnumerable<WeatherForecast> Get()
    {
        _logger.LogInformation("Hi there");

        var rng = new Random();
        return Enumerable.Range(1, 5).Select(index => new WeatherForecast
        {
            Date = DateTime.Now.AddDays(index),
            TemperatureC = rng.Next(-20, 55),
            Summary = Summaries[rng.Next(Summaries.Length)]
        })
        .ToArray();
    }
  }
}
```

Program.cs

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Steeltoe.Extensions.Logging;
```

```
using Steeltoe.Management.Endpoint;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace Eureka.Register.Example
{
   public class Program
   {
      public static void Main(string[] args)
      {
         CreateHostBuilder(args).Build().Run();
      }
      public static IHostBuilder CreateHostBuilder(string[] args) =>
         Host.CreateDefaultBuilder(args)
            .ConfigureWebHostDefaults(webBuilder =>
            {
               webBuilder.UseStartup<Startup>();
            })
            .AddInfoActuator(); // This enables the /actuator/info endpoint
   }
}
```

Startup.cs

```
// This file for your reference, should not modify this file


using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Steeltoe.Management.Tracing;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;


namespace Eureka.Register.Example
{
  public class Startup
  {
    public Startup(IConfiguration configuration)
    {
      Configuration = configuration;
    }
```

```
    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the container.
    public void ConfigureServices(IServiceCollection services)
    {
      services.AddControllers();

      //Steeltoe distributed tracing
      services.AddDistributedTracing();
    }

    // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
      if (env.IsDevelopment())
      {
        app.UseDeveloperExceptionPage();
      }

      app.UseHttpsRedirection();

      app.UseRouting();

      app.UseAuthorization();
```

```
        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllers();
        });
    }
  }
}
```

WeatherForcast.cs

```
// This file for your reference, should not modify this file

using System;

namespace Eureka.Register.Example
{
  public class WeatherForecast
  {
    public DateTime Date { get; set; }

    public int TemperatureC { get; set; }

    public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);
    public string Summary { get; set; }
  }
}
```