

第68期-运行状况检查选项配置

2019年4月21日 01:06

运行状况检查选项设置

筛选运行状况检查

默认情况下，运行健康检查中间件会运行所有已注册的运行状况检查，如果要筛选部分子集，可通过 Tags 进行分组设置。

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddHealthChecks()
        .AddCheck("Foo", () =>
            HealthCheckResult.Healthy("Foo is OK!"), tags: new[] { "foo_tag" })
        .AddCheck("Bar", () =>
            HealthCheckResult.Unhealthy("Bar is unhealthy!"), tags: new[] { "bar_tag" })
        .AddCheck("Baz", () =>
            HealthCheckResult.Healthy("Baz is OK!"), tags: new[] { "baz_tag" });
}

public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    app.UseHealthChecks("/health", new HealthCheckOptions()
    {
        // Filter out the 'Bar' health check. Only Foo and Baz execute.
        Predicate = (check) => check.Tags.Contains("foo_tag") ||
            check.Tags.Contains("baz_tag")
    });
}
```

自定义 HTTP 状态代码

使用 ResultStatusCodes 可自定义运行状况状态到 HTTP 状态代码的映射。以下 StatusCodes 分配是中间件所使用的默认值。更改状态代码值以满足要求。

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    app.UseHealthChecks("/health", new HealthCheckOptions()
    {
        // The following StatusCodes are the default assignments for
        // the HealthStatus properties.
        ResultStatusCodes =
        {
            [HealthStatus.Healthy] = StatusCodes.Status200OK,
            [HealthStatus.Degraded] = StatusCodes.Status200OK,
            [HealthStatus.Unhealthy] = StatusCodes.Status503ServiceUnavailable
        }
    });
}
```

取消响应缓存

```
app.UseHealthChecks("/health", new HealthCheckOptions() { AllowCachingResponses =
```

自定义输出

```
using Microsoft.AspNetCore.Diagnostics.HealthChecks;
using Microsoft.Extensions.Diagnostics.HealthChecks;

public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    app.UseHealthChecks("/health", new HealthCheckOptions()
    {
        // WriteResponse is a delegate used to write the response.
        ResponseWriter = WriteResponse
    });
}

private static Task WriteResponse(HttpContext httpContext,
    HealthReport result)
{
    httpContext.Response.ContentType = "application/json";

    var json = new JObject(
        new JProperty("status", result.Status.ToString()),
        new JProperty("results", new JObject(result.Entries.Select(pair
            new JProperty(pair.Key, new JObject(
                new JProperty("status",
                new JProperty("description",
                new JProperty("data", new
                    p => new JProperty(p.Key,
return httpContext.Response.WriteAsync(
    json.ToString(Formatting.Indented));
}
```

数据库探测

检查数据库是否在正常响应，通过 BeatPulse 库来完成，一般通过 SELECT 1 查询作为判断依

```
Install-Package AspNetCore.HealthChecks.SqlServer -Version 2.2.0
```

```
{
    "ConnectionStrings": {
        "DefaultConnection": "Server=
    }
}
```

```
services.AddHealthChecks().AddSqlServer(Configuration["ConnectionStrings:Default
```

[微软示例](#)