

Router (2): lớp nội bộ, ngoại lệ

Hướng dẫn tự học lập trình C# toàn tập > Router (2): lớp nội bộ, ngoại lệ

Trong bài này chúng ta tiếp tục áp dụng các kỹ thuật để xây dựng lớp Router, bao gồm [nạp chồng toán tử](#), [lớp lồng nhau](#) (nested class), và [ngoại lệ](#).

NỘI DUNG CỦA BÀI [Ấn]

- Thực hành: xây dựng lớp hỗ trợ phân tích truy vấn
 - Bước 1. Tạo class Router
 - Bước 2. Xây dựng lớp Request
- Phân tích code
 - Nested class
 - Ngoại lệ
 - Xử lý ngoại lệ
- Kết luận

Thực hành: xây dựng lớp hỗ trợ phân tích truy vấn

Bước 1. Tạo class Router

Tạo file Router.cs trong thư mục Framework cho lớp Router

```
1. using System;
2. using System.Collections.Generic;
3. using System.Text;
4.
5. namespace Framework // lưu ý không gian tên
6. {
7.     /// <summary>
8.     /// lớp cho phép ánh xạ truy vấn với phương thức
9.     /// </summary>
10.    public class Router
11.    {
12.
13.    }
14. }
```

Bước 2. Xây dựng lớp Request

Bên trong lớp Router xây dựng lớp Request với code như sau:

```
1. using System;
2.
3. namespace Framework
4. {
5.     public class Router
6.     {
7.         /// <summary>
8.         /// lớp xử lý truy vấn
9.         /// </summary>
10.        private class Request
11.        {
12.            /// <summary>
13.            /// thành phần lệnh của truy vấn
14.            /// </summary>
15.            public string Route { get; private set; }
16.
17.            /// <summary>
18.            /// thành phần tham số của truy vấn
19.            /// </summary>
```

```

20.         public Parameter Parameter { get; private set; }
21.
22.         public Request(string request)
23.         {
24.             Analyze(request);
25.         }
26.
27.         /// <summary>
28.         /// phân tích truy vấn để tách ra thành phần lệnh và thành phần tham số
29.         /// </summary>
30.         /// <param name="request"></param>
31.         private void Analyze(string request)
32.         {
33.             // tìm xem trong chuỗi truy vấn có tham số hay không
34.             var firstIndex = request.IndexOf('?');
35.             // trường hợp truy vấn không chứa tham số
36.             if (firstIndex < 0)
37.             {
38.                 Route = request.ToLower().Trim();
39.             }
40.             // trường hợp truy vấn chứa tham số
41.             else
42.             {
43.                 // nếu chuỗi lỗi (chỉ chứa tham số, không chứa route)
44.                 if (firstIndex <= 1) throw new Exception("Invalid request parameter")
45.
46.                 // cắt chuỗi truy vấn lấy mốc là ký tự ?
47.                 // sau phép toán này thu được mảng 2 phần tử: thứ nhất là route, thứ
48.                 var tokens = request.Split(new[] { '?' }, 2, StringSplitOptions.Remove
49.
50.                 // route là thành phần lệnh của truy vấn
51.                 Route = tokens[0].Trim().ToLower();
52.
53.                 // parameter là thành phần tham số của truy vấn
54.                 var parameterPart = request.Substring(firstIndex + 1).Trim();
55.
56.                 Parameter = new Parameter(parameterPart);
57.             }
58.         }
59.     }
60.
61. }
62.

```

Phân tích code

Nested class

Để ý rằng khi xây dựng lớp Router, lớp Request xây dựng **bên trong** lớp Router. Request là một **nested class** của Router.

Ở trường hợp trên, Request còn gọi là lớp trong/lớp nội bộ, Router là lớp ngoài, đồng thời là lớp cấp đỉnh.

Trong lớp Router có 3 loại logic tương đối độc lập: phân tích truy vấn, xử lý chuỗi tham số, đăng ký lệnh và gọi phương thức. Do đó, chúng ta tách Request thành một lớp nội bộ. Do sau này client code sẽ sử dụng đến lớp Parameter, lớp này được tách thành một lớp cùng cấp (lớp sibling) với Router. Trong thân lớp Router chỉ còn chứa code để đăng ký lệnh và gọi phương thức.

Nếu một lớp sibling của Router muốn sử dụng lớp Request (giả sử lớp Request được đánh dấu public) thì phải sử dụng tên lớp là Router.Request trong các lệnh khai báo và khởi tạo, không thể trực tiếp sử dụng trực tiếp tên Request. Tên gọi ngắn gọn Request chỉ có thể sử dụng bên trong lớp Router.

Lưu ý: nên **hạn chế sử dụng lớp lồng nhau**. Việc sử dụng lớp lồng nhau không hợp lý có thể dẫn đến những lỗi khó lường trước, đặc biệt là khi cho lớp trong và lớp ngoài gọi lẫn nhau.

Ngoại lệ

Khi xây dựng lớp `Request` chúng ta gặp một lệnh

```
1. // nếu chuỗi lỗi (chỉ chứa tham số, không chứa route)
2. if (firstIndex <= 1) throw new Exception("Invalid request parameter");
```

Đây là một tình huống đặc biệt trong đó chuỗi truy vấn bị lỗi: người dùng vô tình chỉ nhập phần tham số mà không nhập phần route. Trong trường hợp này chúng ta không có cách gì để xử lý chuỗi truy vấn.

Một số ví dụ khác: khi thực hiện phép chia, nếu mẫu số vô tình nhận giá trị 0, phép chia không thể thực hiện được; khi người dùng yêu cầu truy xuất một file nhưng lại cung cấp sai đường dẫn khiến không thể thực hiện thao tác truy xuất.

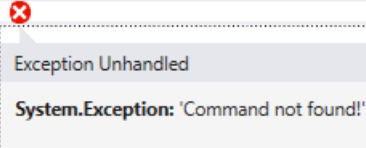
Trong lập trình, những tình huống tương tự xảy ra rất nhiều và được gọi chung là **ngoại lệ** (exception).

Xử lý ngoại lệ

Khi một ngoại lệ được phát ra ở một vị trí bất kỳ trong chương trình, việc thực thi của chương trình sẽ dừng lại. Nếu chương trình đang chạy ở chế độ Debug, trình soạn thảo code sẽ được mở ra và đoạn code bị lỗi sẽ được đánh dấu giúp cho người lập trình xác định vị trí và nguyên nhân gây lỗi.

Hình dưới đây minh họa tình huống lỗi khi người dùng nhập vào một lệnh chưa tồn tại.

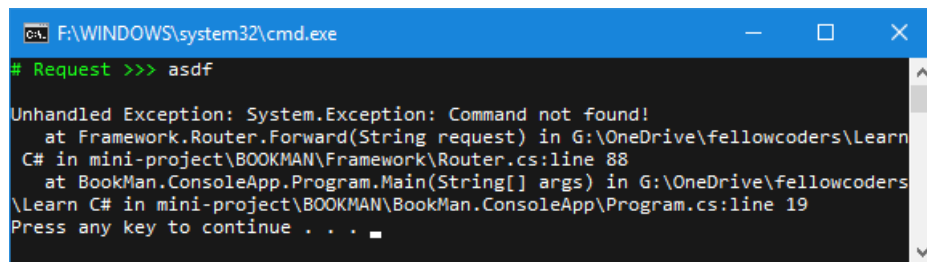
```
var req = new Request(request);
if (!_routingTable.ContainsKey(req.Route))
    throw new Exception("Command not found!");
if (req.Parameter == null)
    _routingTable[req.Route]?.Invoke();
else
    _routingTable[req.Route]?.Invoke(req.Parame
```



Giao diện Visual Studio khi xảy ra ngoại lệ

Nếu chương trình chạy ở chế độ Release, chương trình sẽ bị dừng lại và cơ chế xử lý ngoại lệ mặc định của .NET framework sẽ được kích hoạt để hiển thị lỗi. Chương trình được dịch ở chế độ này sẽ không chạy được ở chế độ Debug nữa.

Nếu chương trình console chạy ở chế độ Release mà gặp lỗi, thông báo lỗi sẽ được hiển thị như dưới đây.



```
F:\WINDOWS\system32\cmd.exe
# Request >>> asdf

Unhandled Exception: System.Exception: Command not found!
   at Framework.Router.Forward(String request) in G:\OneDrive\fellowcoders\Learn C# in mini-project\BOOKMAN\Framework\Router.cs:line 88
   at BookMan.ConsoleApp.Program.Main(String[] args) in G:\OneDrive\fellowcoders\Learn C# in mini-project\BOOKMAN\BookMan.ConsoleApp\Program.cs:line 19
Press any key to continue . . .
```

Thông báo ngoại lệ ở giao diện console

Đây là cơ chế bắt và xử lý lỗi mặc định của .NET framework đối với ứng dụng console. Đối với ứng dụng windows form, giao diện bắt và xử lý lỗi có khác biệt.

Tuy nhiên, cơ chế thông báo lỗi mặc định của .NET framework tương đối không thân thiện với người dùng.

.NET cung cấp cho các chương trình tính năng **bắt và xử lý ngoại lệ** để tự mình xác định xem khi xảy ra lỗi (ngoại lệ) thì sẽ làm gì. Phần xử lý ngoại lệ chúng ta sẽ thực hiện trong bài thực hành cuối cùng.

Kết luận

Trong phần này chúng ta bắt đầu xây dựng các lớp hỗ trợ để phần sau có thể xây dựng lớp Router giúp ánh xạ truy vấn của người dùng sang thực thi phương thức của lớp điều khiển. Qua đây chúng ta đã làm quen với nạp chồng toán tử, phát ra ngoại lệ và lớp lồng nhau.

Trong bài sau chúng ta sẽ tiếp tục sử dụng các class xây dựng trong bài này để hoàn thiện lớp Router.

- + Nếu bạn thấy site hữu ích, trước khi rời đi hãy **giúp đỡ** site bằng một hành động nhỏ để site có thể phát triển và phục vụ bạn tốt hơn.
 - + Nếu bạn thấy bài viết hữu ích, hãy giúp **chia sẻ** tới mọi người.
 - + Nếu có thắc mắc hoặc cần trao đổi thêm, mời bạn viết trong phần **thảo luận** cuối trang.
- Cảm ơn bạn!