遍历树时查找节点

问 9 年零 3 个月前　活跃 9 年零 3 个月前　浏览了 19,000 次

8

1

我想实现一种方法，使我能够在树中找到一个节点。我这样做的方法是递归地使用全局变量来知道何时停止。

我有课：

```
class Node    // represents a node in the tree
{
     // constructor
     public Node() {
          Children = new List<Node>();
     }

     public List<Node> Children;
     public string Name;
     public string Content;
}
```

我现在的方法是：

```
    private bool IsNodeFound = false; // global variable that I use to decide when to
stop

    // method to find a particular node in the tree
    private void Find(Node node, string stringToFind, Action<Node> foundNode)
    {
        if(IsNodeFound)
           return;

        if (node.Content.Contains(stringToFind)){
            foundNode(node);
            IsNodeFound =true;
        }

        foreach (var child in node.Children)
        {
            if (child.Content.Contains(stringToFind)){
                foundNode(node);
                IsNodeFound =true;
            }

            Find(child, stringToFind, foundNode);
        }

    }
```

我使用 Find 方法的方式是这样的：

```
    // root is a node that contain children and those children also contain children
    // root is the "root" of the tree
    IsNodeFound =false;
    Node nodeToFind = null;
```

所以我的问题是如何使这种方法更优雅。我希望方法的签名看起来像：

```
public Node FindNode(Node rootNode);
```

我认为我在做什么是多余的，并且可能有更好的方法来创建该方法。或者也许我可以更改 Node 类，以便我可以使用 linq 查询实现相同的目的。

C# 林克 数据结构 树

分享 改进这个问题 跟随

问2012 年 6 月 22 日 17:49

东野南
30.4k 75 259 426

6 个回答

积极的 | 最老的 | 投票

16

我会这样做：

编写一个实例方法来生成一个节点的子树（如果你不控制这个 Node 类，你可以把它作为一个扩展）：

```
public IEnumerable<Node> GetNodeAndDescendants() // Note that this method is lazy
{
     return new[] { this }
            .Concat(Children.SelectMany(child => child.GetNodeAndDescendants()));
}
```

然后你可以用一点 LINQ 找到节点：

```
var foundNode = rootNode.GetNodeAndDescendants()
                        .FirstOrDefault(node => node.Content.Contains(stringToFind));

if(foundNode != null)
{
    DoSomething(foundNode);
}
```

分享 改进这个答案 跟随

2012 年6 月 22 日 20:11编辑

2012 年6 月 22 日 17:56回答

阿尼
105k 22 245 296

+1 这很棒，因为我可以根据任何标准进行过滤，例如：非常 root.GetSubTree().FirstOrDefault(x => x.Name=="Foo"); 感谢！ —— 东野南 2012 年 6 月 22 日 18:09

如此干净，准确的答案。 —— 安迪英国 2017 年 5 月 3 日 14:39

13

```
public Node Find(string stringToFind)
{
    // find the string, starting with the current instance
    return Find(this, stringToFind);
}

// Search for a string in the specified node and all of its children
public Node Find(Node node, string stringToFind)
{
    if (node.Content.Contains(stringToFind))
        return node;

    foreach (var child in node.Children)
    {
        var result = Find(child, stringToFind);
        if (result != null)
            return result;
    }

    return null;
}
```

分享 改进这个答案 跟随

12 年 6 月 22 日 18:02 回答

瑞安
7,415 2 26 35

您可以使用递归使用深度优先搜索（不需要全局变量知道何时终止）：

3

```
Node FindNode1( Node rootNode, string stringToFind ) {
    if( rootNode.Content == stringToFind ) return rootNode;
    foreach( var child in rootNode.Children ) {
        var n = FindNode1( child, stringToFind );
        if( n != null ) return n;
    }
    return null;
}
```

或者，如果您希望避免递归，您可以使用堆栈以非递归方式完成相同的事情：

```
Node FindNode2( Node rootNode, string stringToFind ) {
    var stack = new Stack<Node>( new[] { rootNode } );
    while( stack.Any() ) {
        var n = stack.Pop();
        if( n.Content == stringToFind ) return n;
        foreach( var child in n.Children ) stack.Push( child );
    }
    return null;
}
```

分享 改进这个答案 跟随

12 年 6 月 22 日 18:06 回答

伊桑·布朗
24.9k 4 72 89

2

```
public Node FindNode(Node rootNode)
{
    if (rootNode.Content.Contains(stringToFind))
       return rootNode;

    foreach (Node node in rootNode.Children)
    {
        if (node.Content.Contains(stringToFind))
           return node;
        else
           return FindNode(node);
    }

    return null;
}
```

分享 改进这个答案 跟随

12 年 6 月 22 日 18:06 回答

凯文·迪特拉利亚

24.5k 17 88 134

1 不能同意更多。虽然在很多情况下基于 LINQ 的解决方案很漂亮，可以简化代码并使其意图非常明确，但我认为这恰恰相反。 —— 尼古拉·阿努塞夫 2012 年 6 月 22 日 18:08

Recursion, and PLinq

1

```
private Node Find(Node node, Func<Node, bool> predicate)
{
    if (predicate(node))
        return node;

    foreach (var n in node.Children.AsParallel())
    {
        var found = Find(n, predicate);
        if (found != default(Node))
            return found;
    }
    return default(Node);
}
```

And calling code:

```
var found = Find(root, (n) => n.Content.Contains("3"));
if (found != default(Node))
    Console.Write("found '{0}'", found.Name);
else Console.Write("not found");
```

Share Improve this answer Follow

answered Jun 22 '12 at 18:24

Juan Ayala

3,190 1 17 23

Consider making LINQ-like API: split "Find" and "Act" parts to make it simple. You may not

```
public IEnumerable<Node> Where(Func<Node, bool> condition);
```

Depending on your needs you may either walk whole tree once and check each node to implement Where, or do it properly with lazy iteration. For lazy iteration you'll need some sort of structure which remembers current position (i.e. stack of nodes to visit and index of child).

Note: please avoid using global variables. I.e. in your current code simply returning true/false from Find function and stopping iteration when true is returned would be better approach.

Share Improve this answer Follow

answered Jun 22 '12 at 17:55

Alexei Levenkov
95.4k 12 116 161