.NET abstract classes

Asked 16 years, 3 months ago Modified 5 years, 10 months ago

Viewed 652 times

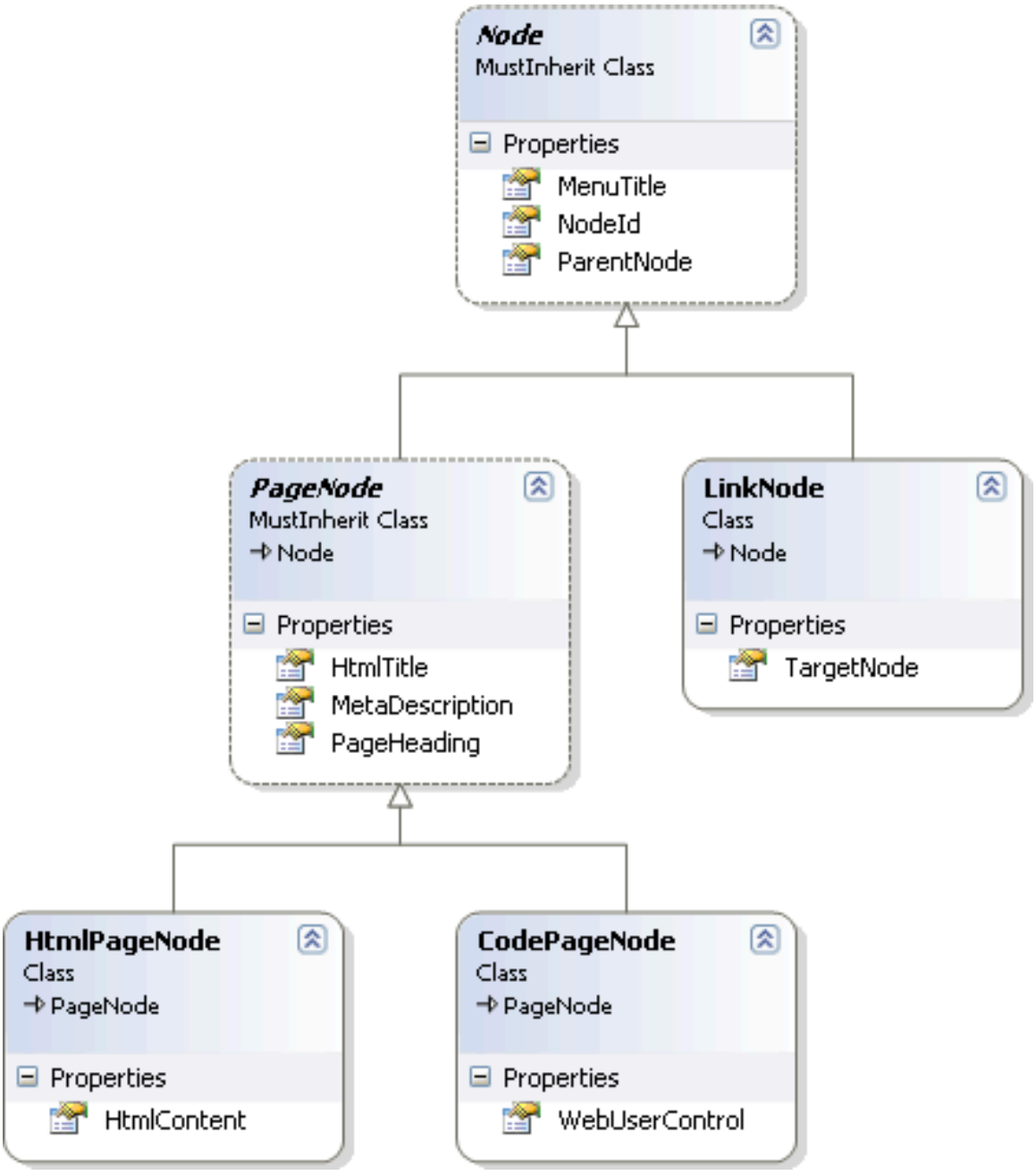
2

I'm designing a web site navigation hierarchy. It's a tree of nodes.

Most nodes are pages. Some nodes are links (think shortcuts in Windows).

Most pages hold HTML content. Some execute code.

I'd like to represent these as this collection of classes and abstract (MustInherit) classes…

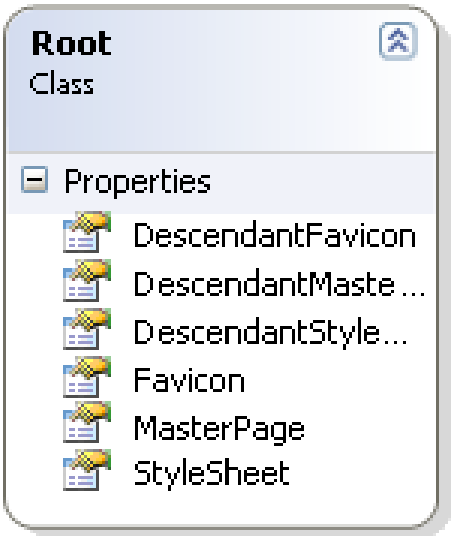


This is the database table where I'm going to store all this…

[database table http://img178.imageshack.us/img178/8573/nodetablefm8.gif](http://img178.imageshack.us/img178/8573/nodetablefm8.gif)

Here's where I'm stumped. PageNodes may or may not be roots.

How should I handle the root class?



I don't want to have to have all four of…

- HtmlPageNode
- CodePageNode
- Html**Root**PageNode
- Code**Root**PageNode

I want the HtmlPageNode and CodePageNode classes to inherit *either* from PageNode or else from RootPageNode. Is that possible?

Clarification: There are multiple root nodes and roots may have parent nodes. Each is the root of only a sub-tree that has distinct styling. Think of different, color-coded departments. (Perhaps root is a poor name choice. Suggestions?)

Update: Regarding the "Root" name...
I've asked: Is there a specific name for the node that coresponds to a subtree?

.net inheritance abstract-class

Share

Improve this question

Follow

edited Dec 30, 2018 at 18:45

Glorfindel

22.6k ● 13 ● 89 ● 116

asked Sep 17, 2008 at 15:09

Zack Peterson

57.3k ● 80 ● 210 ● 280

6 Answers

Sorted by: Highest score (default)

2

As noted, the Composite Pattern may be a good solution.

If that doesn't work for you, it may be simpler - if appropriate - to define Root as an Interface, and apply that as needed.

Of course, that doesnt let you provide any implementation for Root...
If Root must have implementation, you can use the Decorator Pattern.

Share Improve this answer

Follow

answered Sep 17, 2008 at 16:18

AviD

13.1k ● 7 ● 64 ● 93

Use the the Composite Pattern.

2

With regard to your root nodes, are there differences in functionality or is the difference it entirely appearance? If the difference is appearance only I suggest you have an association with a separate Style class from your PageNode.

If there are differences in functionality AND you have lots of types of page then think about using the Decorator Pattern.

Share Improve this answer Follow

edited Sep 18, 2008 at 8:59

answered Sep 17, 2008 at 15:26

Martin Spamer

5,575 ● 30 ● 44

I need clarification. See my answer. – Zack Peterson Sep 17, 2008 at 16:14

1

> I want the HtmlPageNode and CodePageNode classes to inherit either from PageNode or else from RootPageNode. Is that possible?

Yeah it's possible. You need to have HtmlPageNode and codePageNode have an object that will be an Abstract class that PageNode will inherit and RootPageNode too. In the constructor of HtmlPageNode and codePageNode you accept your new Abstract class that will be in your

case PageNode OR RootPageNode. This way you have 2 differents classes with same methods but two differents object. Hope that help you!

Share Improve this answer

Follow

answered Sep 17, 2008 at 15:31

Patrick Desjardins

141k ● 89 ● 294 ● 346

1

Actually, as the "root" node is a special case of node, maybe you need RootHtmlPageNode : HtmlPageNode.

Another idea: as you do not specify what is the difference between a "root" and normal node, maybe just a flag in node specifying if it is root or not also will be a good design.

EDIT: Per your clarification, there is no functional difference between normal and root node, so a simple flag should be enough (or property IsRootNode). If "root" node only supplies a styling data (or any other data for itself and it's children), then you can place this styling data in a separate structure/class and fetch it recursively (based on IsRootNode):

```
class Node
{
   private bool isRootNode;
   public bool IsRootNode;

   private StylingData stylingData;
   public StylingData StylingData
   {
      set
```

```
        {
            if (this.IsRootNode)
                this.stylingData = value;
            else
                throw new ApplicationException("The node i
        }
        get
        {
            if (this.IsRootNode)
                return this.stylingData;
            else
                return this.parent.StylingData;
        }
    }
}
```

This assumes, that each node has a reference to it's parent.

It become way beyond the question, as I do not know the exact design.

Share Improve this answer Follow

edited Sep 17, 2008 at 19:48

answered Sep 17, 2008 at 15:16

Sunny Milenov

22.3k ● 6 ● 82 ● 107

1

Clarification: There are multiple root nodes and roots may have parent nodes. Each is the root of only a sub-tree that has distinct styling. Think of

> different, color-coded departments. (Perhaps root is a poor name choice. Suggestions?)

Root is a poor name choice because it's (somewhat ironically) accepted as explicitly the top level of a tree structure, because the tree starts where root comes out of the ground. Any node beyond that is a branch or leaf and not directly attached to the root.

A better name would be something like IsAuthoritativeStyleNode, IsCascadingStyleNode, IsStyleParentNode or instead qualify it: e.g. IsDepartmentRootNode. Giving things clear unambiguous names is one of things that drastically improves readability / easy understanding.

You can't really achieve what you want just via abstract base classes/inheritence. As per other suggestion(s), consider interfaces instead.

I'd also consider thinking about whether you're letting the database schema drive your client side class design too much. Not saying it needs changing in this case, but it should at least be thought about. Think about how you could factor out properties into separate tables referencing the common 'Node' table, and normalize them to minimize nulls and/or duplicated identical data.

edited Sep 18, 2008 at 5:57

answered Sep 17, 2008 at 17:18

Gordon Hartley

489 ● 4 ● 10

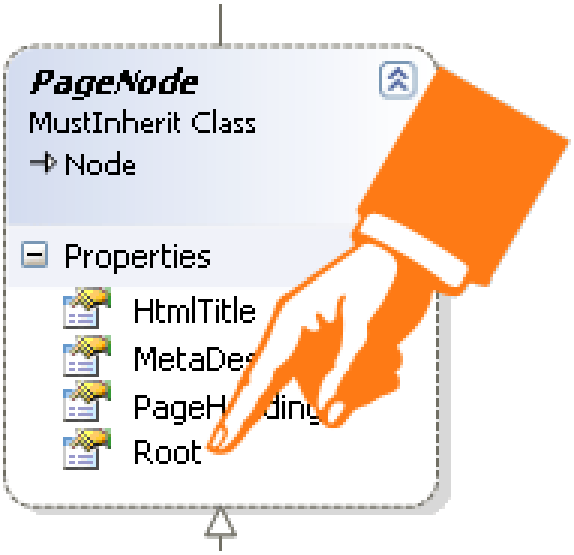
I'm still trying to zero in on a final architecture. I initially started with many small tables with almost zero nullalbe fields. stackoverflow.com/questions/56981/… – Zack Peterson Sep 17, 2008 at 18:08

And I learned about the LINQ to SQL IsDiscriminator property. stackoverflow.com/questions/57152/… – Zack Peterson Sep 17, 2008 at 18:09

But then I learned that LINQ to SQL only supports single table inheritence. stackoverflow.com/questions/67659/… – Zack Peterson Sep 17, 2008 at 18:09

0

Should the PageNode class simply have a property of type Root?



Is that counter to the idea that a PageNode is-a Root. Or, are they not "is-a Root" because only *some* of them are roots?

And does that imply that the property might traverse the tree looking for the root ancestor? Or is that just me?

Share Improve this answer

Follow

edited Feb 18, 2019 at 7:19

Glorfindel

22.6k ● 13 ● 89 ● 116

answered Sep 17, 2008 at 15:55

Zack Peterson

57.3k ● 80 ● 210 ● 280

Dont get too hung up on the is-a or has-a discussion... If one of them doesnt "feel" automatically right, then it doesnt matter, and just go with what works. – AviD Sep 18, 2008 at 21:06