**语法树**

语法树是用于编译、代码分析、绑定、重构、IDE 功能和代码生成的主要结构。

语法树具有三个关键特性。

1. 是语法树完全保真地保留所有源代码信息，即通过语法树可以看到原来的代码
2. 编辑语法树会创建新的树，而不会更改现有树，通过这些操作，可高效编辑文本。
3. 获取的树是代码当前状态的快照，不会更改。 这可让多个用户同时在不同线程中与同一语法树进行交互，而不会锁定或重复

语法树实际上是一个树形数据结构，其中非终端结构化元素是其他元素的父元素。

**语法节点**

语法树由语法节点组成。如声明、语句、子句和表达式等都是语法节点。 语法节点派生自 Microsoft.CodeAnalysis.SyntaxNode 类。

所有语法节点都是语法树中的非终端节点，每个节点都具有可通过 SyntaxNode.Parent 属性访问的父节点。

每个节点都包含 SyntaxNode.ChildNodes() 方法，返回所有子节点。

此外，每个语法节点子类通过强类型属性公开所有相同的子级。 例如，BinaryExpressionSyntax 节点类具有三个特定于二元运算符的其他属性：Left、OperatorToken 和 Right。 Left 和 Right 的类型为 ExpressionSyntax，OperatorToken 的类型为 SyntaxToken。

**语法标记（SyntaxToken）**

语法标记是语言语法的终端，表示代码的最小语法片段。它们从不作为其他节点或标记的父级。 语法标记包含关键字、标识符、文本和标点，如static，int等。

ValueText 属性是其文本

**语法琐碎内容**

语法琐碎内容表示对正常理解代码基本上没有意义的源文本部分，例如空格、注释和预处理器指令。单个 Microsoft.CodeAnalysis.SyntaxTrivia 类型用于描述各种类型的琐碎内容。

可通过检查标记的 SyntaxToken.LeadingTrivia 或 SyntaxToken.TrailingTrivia 集合来访问琐碎内容。

每个琐碎内容都与一个标记关联，因此可使用 SyntaxTrivia.Token 属性访问关联的标记。

**范围**

每个节点、标记或琐碎内容都知道其在源文本内的位置和包含的字符数。

TextSpan表示一个文本范围

每个节点具有两个 TextSpan 属性：Span 和 FullSpan。

Span 属性：

表示从节点子树中第一个标记的开头到最后一个标记末尾的文本范围。此范围不包括任何前导或尾随琐碎内容。

FullSpan 属性：与Span相同，但包括前导或尾随琐碎内容的范围。