Roslyn 笔记

作者：xu365082218

原文链接：https://ld246.com/article/1501926025547

来源网站：链滴

最近发现一些脚本语言很有意思，很想写一门新的脚本语言，所以找到了 C#的语法分析相关的
容
using Microsoft.CodeAnalysis.CSharp.Syntax;
using Microsoft.CodeAnalysis.CSharp;
namespace Microsoft.CodeAnalysis.CSharp
{
public enum SyntaxKind : ushort
{
None = 0,
List = 1,
TildeToken = 8193,
ExclamationToken = 8194,
DollarToken = 8195,
PercentToken = 8196,
CaretToken = 8197,
AmpersandToken = 8198,
AsteriskToken = 8199,
OpenParenToken = 8200,
CloseParenToken = 8201,
MinusToken = 8202,
PlusToken = 8203,
EqualsToken = 8204,
OpenBraceToken = 8205,
CloseBraceToken = 8206,
OpenBracketToken = 8207,
CloseBracketToken = 8208,
BarToken = 8209,
BackslashToken = 8210,
ColonToken = 8211,
SemicolonToken = 8212,
DoubleQuoteToken = 8213,
SingleQuoteToken = 8214,
LessThanToken = 8215,
CommaToken = 8216,
GreaterThanToken = 8217,
DotToken = 8218,
QuestionToken = 8219,
HashToken = 8220,
SlashToken = 8221,
SlashGreaterThanToken = 8232,
LessThanSlashToken = 8233,
XmlCommentStartToken = 8234,
XmlCommentEndToken = 8235,
XmlCDataStartToken = 8236,
XmlCDataEndToken = 8237,
XmlProcessingInstructionStartToken = 8238,
XmlProcessingInstructionEndToken = 8239,
BarBarToken = 8260,
AmpersandAmpersandToken = 8261,
MinusMinusToken = 8262,
PlusPlusToken = 8263,
ColonColonToken = 8264,
QuestionQuestionToken = 8265,
MinusGreaterThanToken = 8266,

ExclamationEqualsToken = 8267,
EqualsEqualsToken = 8268,
EqualsGreaterThanToken = 8269,
LessThanEqualsToken = 8270,
LessThanLessThanToken = 8271,
LessThanLessThanEqualsToken = 8272,
GreaterThanEqualsToken = 8273,
GreaterThanGreaterThanToken = 8274,
GreaterThanGreaterThanEqualsToken = 8275,
SlashEqualsToken = 8276,
AsteriskEqualsToken = 8277,
BarEqualsToken = 8278,
AmpersandEqualsToken = 8279,
PlusEqualsToken = 8280,
MinusEqualsToken = 8281,
CaretEqualsToken = 8282,
PercentEqualsToken = 8283,
BoolKeyword = 8304,
ByteKeyword = 8305,
SByteKeyword = 8306,
ShortKeyword = 8307,
UShortKeyword = 8308,
IntKeyword = 8309,
UIntKeyword = 8310,
LongKeyword = 8311,
ULongKeyword = 8312,
DoubleKeyword = 8313,
FloatKeyword = 8314,
DecimalKeyword = 8315,
StringKeyword = 8316,
CharKeyword = 8317,
VoidKeyword = 8318,
ObjectKeyword = 8319,
TypeOfKeyword = 8320,
SizeOfKeyword = 8321,
NullKeyword = 8322,
TrueKeyword = 8323,
FalseKeyword = 8324,
IfKeyword = 8325,
ElseKeyword = 8326,
WhileKeyword = 8327,
ForKeyword = 8328,
ForEachKeyword = 8329,
DoKeyword = 8330,
SwitchKeyword = 8331,
CaseKeyword = 8332,
DefaultKeyword = 8333,
TryKeyword = 8334,
CatchKeyword = 8335,
FinallyKeyword = 8336,
LockKeyword = 8337,
GotoKeyword = 8338,
BreakKeyword = 8339,
ContinueKeyword = 8340,

ReturnKeyword = 8341,
ThrowKeyword = 8342,
PublicKeyword = 8343,
PrivateKeyword = 8344,
InternalKeyword = 8345,
ProtectedKeyword = 8346,
StaticKeyword = 8347,
ReadOnlyKeyword = 8348,
SealedKeyword = 8349,
ConstKeyword = 8350,
FixedKeyword = 8351,
StackAllocKeyword = 8352,
VolatileKeyword = 8353,
NewKeyword = 8354,
OverrideKeyword = 8355,
AbstractKeyword = 8356,
VirtualKeyword = 8357,
EventKeyword = 8358,
ExternKeyword = 8359,
RefKeyword = 8360,
OutKeyword = 8361,
InKeyword = 8362,
IsKeyword = 8363,
AsKeyword = 8364,
ParamsKeyword = 8365,
ArgListKeyword = 8366,
MakeRefKeyword = 8367,
RefTypeKeyword = 8368,
RefValueKeyword = 8369,
ThisKeyword = 8370,
BaseKeyword = 8371,
NamespaceKeyword = 8372,
UsingKeyword = 8373,
ClassKeyword = 8374,
StructKeyword = 8375,
InterfaceKeyword = 8376,
EnumKeyword = 8377,
DelegateKeyword = 8378,
CheckedKeyword = 8379,
UncheckedKeyword = 8380,
UnsafeKeyword = 8381,
OperatorKeyword = 8382,
ExplicitKeyword = 8383,
ImplicitKeyword = 8384,
YieldKeyword = 8405,
PartialKeyword = 8406,
AliasKeyword = 8407,
GlobalKeyword = 8408,
AssemblyKeyword = 8409,
ModuleKeyword = 8410,
TypeKeyword = 8411,
FieldKeyword = 8412,
MethodKeyword = 8413,
ParamKeyword = 8414,

PropertyKeyword = 8415,
TypeVarKeyword = 8416,
GetKeyword = 8417,
SetKeyword = 8418,
AddKeyword = 8419,
RemoveKeyword = 8420,
WhereKeyword = 8421,
FromKeyword = 8422,
GroupKeyword = 8423,
JoinKeyword = 8424,
IntoKeyword = 8425,
LetKeyword = 8426,
ByKeyword = 8427,
SelectKeyword = 8428,
OrderByKeyword = 8429,
OnKeyword = 8430,
EqualsKeyword = 8431,
AscendingKeyword = 8432,
DescendingKeyword = 8433,
NameOfKeyword = 8434,
AsyncKeyword = 8435,
AwaitKeyword = 8436,
ElifKeyword = 8467,
EndIfKeyword = 8468,
RegionKeyword = 8469,
EndRegionKeyword = 8470,
DefineKeyword = 8471,
UndefKeyword = 8472,
WarningKeyword = 8473,
ErrorKeyword = 8474,
LineKeyword = 8475,
PragmaKeyword = 8476,
HiddenKeyword = 8477,
ChecksumKeyword = 8478,
DisableKeyword = 8479,
RestoreKeyword = 8480,
ReferenceKeyword = 8481,
OmittedTypeArgumentToken = 8492,
OmittedArraySizeExpressionToken = 8493,
EndOfDirectiveToken = 8494,
EndOfDocumentationCommentToken = 8495,
EndOfFileToken = 8496,
BadToken = 8507,
IdentifierToken = 8508,
NumericLiteralToken = 8509,
CharacterLiteralToken = 8510,
StringLiteralToken = 8511,
XmlEntityLiteralToken = 8512,
XmlTextLiteralToken = 8513,
XmlTextLiteralNewLineToken = 8514,
InterpolatedStringToken = 8515,
InterpolatedStringStartToken = 8516,
InterpolatedStringMidToken = 8517,
InterpolatedStringEndToken = 8518,

EndOfLineTrivia = 8539,
WhitespaceTrivia = 8540,
SingleLineCommentTrivia = 8541,
MultiLineCommentTrivia = 8542,
DocumentationCommentExteriorTrivia = 8543,
SingleLineDocumentationCommentTrivia = 8544,
MultiLineDocumentationCommentTrivia = 8545,
DisabledTextTrivia = 8546,
PreprocessingMessageTrivia = 8547,
IfDirectiveTrivia = 8548,
ElifDirectiveTrivia = 8549,
ElseDirectiveTrivia = 8550,
EndIfDirectiveTrivia = 8551,
RegionDirectiveTrivia = 8552,
EndRegionDirectiveTrivia = 8553,
DefineDirectiveTrivia = 8554,
UndefDirectiveTrivia = 8555,
ErrorDirectiveTrivia = 8556,
WarningDirectiveTrivia = 8557,
LineDirectiveTrivia = 8558,
PragmaWarningDirectiveTrivia = 8559,
PragmaChecksumDirectiveTrivia = 8560,
ReferenceDirectiveTrivia = 8561,
BadDirectiveTrivia = 8562,
SkippedTokensTrivia = 8563,
XmlElement = 8574,
XmlElementStartTag = 8575,
XmlElementEndTag = 8576,
XmlEmptyElement = 8577,
XmlTextAttribute = 8578,
XmlCrefAttribute = 8579,
XmlNameAttribute = 8580,
XmlName = 8581,
XmlPrefix = 8582,
XmlText = 8583,
XmlCDataSection = 8584,
XmlComment = 8585,
XmlProcessingInstruction = 8586,
TypeCref = 8597,
QualifiedCref = 8598,
NameMemberCref = 8599,
IndexerMemberCref = 8600,
OperatorMemberCref = 8601,
ConversionOperatorMemberCref = 8602,
CrefParameterList = 8603,
CrefBracketedParameterList = 8604,
CrefParameter = 8605,
IdentifierName = 8616,
QualifiedName = 8617,
GenericName = 8618,
TypeArgumentList = 8619,
AliasQualifiedName = 8620,
PredefinedType = 8621,
ArrayType = 8622,

ArrayRankSpecifier = 8623,
PointerType = 8624,
NullableType = 8625,
OmittedTypeArgument = 8626,
ParenthesizedExpression = 8632,
ConditionalExpression = 8633,
InvocationExpression = 8634,
ElementAccessExpression = 8635,
ArgumentList = 8636,
BracketedArgumentList = 8637,
Argument = 8638,
NameColon = 8639,
CastExpression = 8640,
AnonymousMethodExpression = 8641,
SimpleLambdaExpression = 8642,
ParenthesizedLambdaExpression = 8643,
ObjectInitializerExpression = 8644,
CollectionInitializerExpression = 8645,
ArrayInitializerExpression = 8646,
AnonymousObjectMemberDeclarator = 8647,
ComplexElementInitializerExpression = 8648,
ObjectCreationExpression = 8649,
AnonymousObjectCreationExpression = 8650,
ArrayCreationExpression = 8651,
ImplicitArrayCreationExpression = 8652,
StackAllocArrayCreationExpression = 8653,
OmittedArraySizeExpression = 8654,
InterpolatedString = 8655,
ImplicitElementAccess = 8656,
AddExpression = 8668,
SubtractExpression = 8669,
MultiplyExpression = 8670,
DivideExpression = 8671,
ModuloExpression = 8672,
LeftShiftExpression = 8673,
RightShiftExpression = 8674,
LogicalOrExpression = 8675,
LogicalAndExpression = 8676,
BitwiseOrExpression = 8677,
BitwiseAndExpression = 8678,
ExclusiveOrExpression = 8679,
EqualsExpression = 8680,
NotEqualsExpression = 8681,
LessThanExpression = 8682,
LessThanOrEqualExpression = 8683,
GreaterThanExpression = 8684,
GreaterThanOrEqualExpression = 8685,
IsExpression = 8686,
AsExpression = 8687,
CoalesceExpression = 8688,
SimpleMemberAccessExpression = 8689,
PointerMemberAccessExpression = 8690,
ConditionalAccessExpression = 8691,
MemberBindingExpression = 8707,

ElementBindingExpression = 8708,
SimpleAssignmentExpression = 8714,
AddAssignmentExpression = 8715,
SubtractAssignmentExpression = 8716,
MultiplyAssignmentExpression = 8717,
DivideAssignmentExpression = 8718,
ModuloAssignmentExpression = 8719,
AndAssignmentExpression = 8720,
ExclusiveOrAssignmentExpression = 8721,
OrAssignmentExpression = 8722,
LeftShiftAssignmentExpression = 8723,
RightShiftAssignmentExpression = 8724,
UnaryPlusExpression = 8730,
UnaryMinusExpression = 8731,
BitwiseNotExpression = 8732,
LogicalNotExpression = 8733,
PreIncrementExpression = 8734,
PreDecrementExpression = 8735,
PointerIndirectionExpression = 8736,
AddressOfExpression = 8737,
PostIncrementExpression = 8738,
PostDecrementExpression = 8739,
AwaitExpression = 8740,
ThisExpression = 8746,
BaseExpression = 8747,
ArgListExpression = 8748,
NumericLiteralExpression = 8749,
StringLiteralExpression = 8750,
CharacterLiteralExpression = 8751,
TrueLiteralExpression = 8752,
FalseLiteralExpression = 8753,
NullLiteralExpression = 8754,
TypeOfExpression = 8760,
SizeOfExpression = 8761,
CheckedExpression = 8762,
UncheckedExpression = 8763,
DefaultExpression = 8764,
MakeRefExpression = 8765,
RefValueExpression = 8766,
RefTypeExpression = 8767,
NameOfExpression = 8768,
QueryExpression = 8774,
QueryBody = 8775,
FromClause = 8776,
LetClause = 8777,
JoinClause = 8778,
JoinIntoClause = 8779,
WhereClause = 8780,
OrderByClause = 8781,
AscendingOrdering = 8782,
DescendingOrdering = 8783,
SelectClause = 8784,
GroupClause = 8785,
QueryContinuation = 8786,

Block = 8792,
LocalDeclarationStatement = 8793,
VariableDeclaration = 8794,
VariableDeclarator = 8795,
EqualsValueClause = 8796,
ExpressionStatement = 8797,
EmptyStatement = 8798,
LabeledStatement = 8799,
GotoStatement = 8800,
GotoCaseStatement = 8801,
GotoDefaultStatement = 8802,
BreakStatement = 8803,
ContinueStatement = 8804,
ReturnStatement = 8805,
YieldReturnStatement = 8806,
YieldBreakStatement = 8807,
ThrowStatement = 8808,
WhileStatement = 8809,
DoStatement = 8810,
ForStatement = 8811,
ForEachStatement = 8812,
UsingStatement = 8813,
FixedStatement = 8814,
CheckedStatement = 8815,
UncheckedStatement = 8816,
UnsafeStatement = 8817,
LockStatement = 8818,
IfStatement = 8819,
ElseClause = 8820,
SwitchStatement = 8821,
SwitchSection = 8822,
CaseSwitchLabel = 8823,
DefaultSwitchLabel = 8824,
TryStatement = 8825,
CatchClause = 8826,
CatchDeclaration = 8827,
CatchFilterClause = 8828,
FinallyClause = 8829,
CompilationUnit = 8840,
GlobalStatement = 8841,
NamespaceDeclaration = 8842,
UsingDirective = 8843,
ExternAliasDirective = 8844,
AttributeList = 8847,
AttributeTargetSpecifier = 8848,
Attribute = 8849,
AttributeArgumentList = 8850,
AttributeArgument = 8851,
NameEquals = 8852,
ClassDeclaration = 8855,
StructDeclaration = 8856,
InterfaceDeclaration = 8857,
EnumDeclaration = 8858,
DelegateDeclaration = 8859,

BaseList = 8865,
TypeParameterConstraintClause = 8866,
ConstructorConstraint = 8867,
ClassConstraint = 8868,
StructConstraint = 8869,
TypeConstraint = 8870,
ExplicitInterfaceSpecifier = 8871,
EnumMemberDeclaration = 8872,
FieldDeclaration = 8873,
EventFieldDeclaration = 8874,
MethodDeclaration = 8875,
OperatorDeclaration = 8876,
ConversionOperatorDeclaration = 8877,
ConstructorDeclaration = 8878,
BaseConstructorInitializer = 8889,
ThisConstructorInitializer = 8890,
DestructorDeclaration = 8891,
PropertyDeclaration = 8892,
EventDeclaration = 8893,
IndexerDeclaration = 8894,
AccessorList = 8895,
GetAccessorDeclaration = 8896,
SetAccessorDeclaration = 8897,
AddAccessorDeclaration = 8898,
RemoveAccessorDeclaration = 8899,
UnknownAccessorDeclaration = 8900,
ParameterList = 8906,
BracketedParameterList = 8907,
Parameter = 8908,
TypeParameterList = 8909,
TypeParameter = 8910,
IncompleteMember = 8916,
ArrowExpressionClause = 8917,
InterpolatedStringInsert = 8918
}
}
每一种 SyntaxNode node 都有一个 CSharpKind()表明自身的节点语法类型，而每一种节点语法类，往往有一个相关的类来处理分析该语法类型的语法实例的参数和一些相关变量
类似如:Class 类型定义
SyntaxKind.ClassDeclaration 类型的语法节点，在 namespace Microsoft.CodeAnalysis.CSharp.Syntax 里有 ClassDeclarationSyntax 类型处理解析此语法节点
如何得到一个文件的语法树呢
SyntaxTree tree = CSharpSyntaxTree.ParseText("代码内容");//代码内容就是代码文件里的内容，者直接的文本值
tree.GetRoot()就是语法树根节点
tree.GetRoot().ChildNodes()就是子节点，每一个子节点都是一颗语法子树
这样遍历语法树，就可以得到每一个语法节点的任意数据。
当控制了一个文件形成的语法树的全部语法节点时，那么就可以用这些信息做一些事情
做什么事情
1：在 C#工程里集成 Roslyn 编译器使得 C#自解释自己[这是一个想法，就是在 C#工程里集成 C#编器 Roslyn 然后加载文本文件作为脚本，与原生工程里的 C#代码集成使用，也就是互相类型透明] 这想法比较前端，未来肯定有这种技术，就是语言自身既是编译语言，也是解释语言，既可以编译成字码使用，还可以调用外部自身语言语法规范的脚本，在 C#中 Microsoft.CodeAnalysis.Scripting 有这方面的一些实现，不过后来好像移除了，这个部分也可以看出微软对这方面曾经有发力

2：写一些代码转换工具，大部分针对 Java 和 C#简单语法互相转换的.或者是拥有大量类成员的类的相转换
3：可以仿造 C#的语法树类型以及各种类型的参数，来实现自己的语法树系统，这部分就与设计新的本语言类似，不过细节肯定是非常复杂的
题外话
人类语言就中国来说，普通话就是北京话，而程序语言似乎并没有一种语言可以把所有语言联通起来在能使用一个程序语言的环境里，大部分情况下，调用其他程序语言，往往比较复杂，类似调用 dll 或者使用脚本语言，等
为什么不能做到一个工程里，几个 C 文件几个 C++ 文件几个 C#文件然后各按各自的编译然后生成。