ChilliCream / **graphql-platform** Public

Code | Issues 287 | Pull requests 89 | Discussions | Actions | Projects

Commit b9271e6

michaelstaib committed last week · ✕ 0 / 1

```
Add depth limit to GraphQL parser
```

main-version-12 · 12.22.7

1 parent 386abb3 commit b9271e6

10 files changed +116 -22 lines changed

Top

Filter files…

- src/HotChocolate/Language/src/Language.Utf8
  - ParserOptions.cs
  - Properties
    - LangUtf8Resources.Designer.cs
    - LangUtf8Resources.resx
  - Utf8GraphQLParser.Directives.cs
  - Utf8GraphQLParser.Fragments.cs
  - Utf8GraphQLParser.Operations.cs
  - Utf8GraphQLParser.Types.cs
  - Utf8GraphQLParser.Utilities.cs
  - Utf8GraphQLParser.Values.cs
  - Utf8GraphQLParser.cs

10 files changed +116 -22 lines changed

Search within code

…Language/src/Language.Utf8/ParserOptions.cs

```
@@ -21,8 +21,37 @@ public ParserOptions(
21 21            NoLocations = noLocations;
22 22            Experimental = new ParserOptionsExperimental(
23 23                allowFragmentVariables);
   24 +          MaxAllowedDirectives = 4;
   25 +          MaxAllowedRecursionDepth = 200;
24 26        }
25 27
   28 +    /// <summary>
   29 +    /// Initializes a new instance of <see cref="ParserOptions"/> with security limits.
   30 +    /// </summary>
   31 +    public ParserOptions(
   32 +        bool noLocations,
   33 +        bool allowFragmentVariables,
   34 +        int maxAllowedDirectives,
   35 +        int maxAllowedRecursionDepth)
   36 +    {
   37 +        NoLocations = noLocations;
   38 +        Experimental = new ParserOptionsExperimental(allowFragmentVariables);
   39 +        MaxAllowedDirectives = maxAllowedDirectives;
   40 +        MaxAllowedRecursionDepth = maxAllowedRecursionDepth;
   41 +    }
   42 +
   43 +    /// <summary>
   44 +    /// The maximum number of directives allowed per location (e.g. per field,
   45 +    /// per operation, per fragment definition). Repeatable directives can be used
   46 +    /// to exhaust CPU and memory resources if not limited.
   47 +    /// </summary>
   48 +    public int MaxAllowedDirectives { get; }
   49 +
   50 +    /// <summary>
   51 +    /// Gets the maximum allowed recursion depth of a parsed document.
   52 +    /// </summary>
   53 +    public int MaxAllowedRecursionDepth { get; }
   54 +
26 55        /// <summary>
27 56        /// By default, the parser creates <see cref="ISyntaxNode" />s
28 57        /// that know the location in the source that they correspond to.
```

...f8/Properties/LangUtf8Resources.Designer.cs

Load Diff

Some generated files are not rendered by default. Learn more about customizing how changed files appear on GitHub.

...uage.Utf8/Properties/LangUtf8Resources.resx

```
@@ -75,4 +75,10 @@
75  75    <data name="UnexpectedToken" xml:space="preserve">
76  76      <value>Unexpected token: {0}.</value>
77  77    </data>
    78  +   <data name="Utf8GraphQLParser_ParseDirective_MaxAllowedDirectivesReached" xml:space="preserve">
    79  +     <value>A location in the GraphQL document contains more than {0} directives. Parsing aborted.</value>
    80  +   </data>
    81  +   <data name="Utf8GraphQLParser_Start_MaxAllowedRecursionDepthReached" xml:space="preserve">
    82  +     <value>Document exceeds the maximum allowed recursion depth of {0}. Parsing aborted.</value>
    83  +   </data>
78  84  </root>
```

...nguage.Utf8/Utf8GraphQLParser.Directives.cs

```
@@ -1,5 +1,6 @@
1   1   using System.Collections.Generic;
2   2   using System.Runtime.CompilerServices;
    3   + using static HotChocolate.Language.Properties.LangUtf8Resources;
3   4
4   5   namespace HotChocolate.Language;
5   6
@@ -67,7 +68,7 @@ private NameNode ParseDirectiveLocation()
67  68              throw Unexpected(kind);
68  69          }
```

```
69  70
70      -     private List<DirectiveNode> ParseDirectives(bool isConstant)
        71  +     private List<DirectiveNode> ParseDirectives(bool isConstant, bool isQueryLocation = false)
71  72          {
72  73              if (_reader.Kind == TokenKind.At)
73  74              {
        @@ -76,6 +77,15 @@ private List<DirectiveNode> ParseDirectives(bool isConstant)
76  77                  while (_reader.Kind == TokenKind.At)
77  78                  {
78  79                      list.Add(ParseDirective(isConstant));
        80  +
        81  +                 if (isQueryLocation && list.Count > _maxAllowedDirectives)
        82  +                 {
        83  +                     throw new SyntaxException(
        84  +                         _reader,
        85  +                         string.Format(
        86  +                             Utf8GraphQLParser_ParseDirective_MaxAllowedDirectivesReached,
        87  +                             _maxAllowedDirectives));
        88  +                 }
79  89              }
80  90
81  91              return list;
```

...anguage.Utf8/Utf8GraphQLParser.Fragments.cs

```
        @@ -58,7 +58,7 @@ private FragmentDefinitionNode ParseFragmentDefinition()
58  58                    ParseVariableDefinitions();
59  59                ExpectOnKeyword();
60  60                NamedTypeNode typeCondition = ParseNamedType();
61      -             List<DirectiveNode> directives = ParseDirectives(false);
        61  +             List<DirectiveNode> directives = ParseDirectives(false, isQueryLocation: true);
62  62                SelectionSetNode selectionSet = ParseSelectionSet();
63  63                Location? location = CreateLocation(in start);
64  64
        @@ -77,7 +77,7 @@ private FragmentDefinitionNode ParseFragmentDefinition()
77  77                NameNode name = ParseFragmentName();
```

```
78  78              ExpectOnKeyword();
79  79              NamedTypeNode typeCondition = ParseNamedType();
80      -           List<DirectiveNode> directives = ParseDirectives(false);
    80  +           List<DirectiveNode> directives = ParseDirectives(false,
                    isQueryLocation: true);
81  81              SelectionSetNode selectionSet = ParseSelectionSet();
82  82              Location? location = CreateLocation(in start);
83  83
    @@ -104,7 +104,7 @@ private FragmentDefinitionNode ParseFragmentDefinition()
104 104         private FragmentSpreadNode ParseFragmentSpread(in TokenInfo start)
105 105         {
106 106             NameNode name = ParseFragmentName();
107     -           List<DirectiveNode> directives = ParseDirectives(false);
    107 +           List<DirectiveNode> directives = ParseDirectives(false,
                    isQueryLocation: true);
108 108             Location? location = CreateLocation(in start);
109 109
110 110             return new FragmentSpreadNode
    @@ -130,7 +130,7 @@ private InlineFragmentNode ParseInlineFragment(
130 130             in TokenInfo start,
131 131             NamedTypeNode? typeCondition)
132 132         {
133     -           List<DirectiveNode> directives = ParseDirectives(false);
    133 +           List<DirectiveNode> directives = ParseDirectives(false,
                    isQueryLocation: true);
134 134             SelectionSetNode selectionSet = ParseSelectionSet();
135 135             Location? location = CreateLocation(in start);
136 136
```

...nguage.Utf8/Utf8GraphQLParser.Operations.cs

```
    @@ -25,7 +25,7 @@ private OperationDefinitionNode ParseOperationDefinition()
25  25              OperationType operation = ParseOperationType();
26  26              NameNode? name = _reader.Kind == TokenKind.Name ? ParseName() : null;
27  27              List<VariableDefinitionNode> variableDefinitions =
                    ParseVariableDefinitions();
28      -           List<DirectiveNode> directives = ParseDirectives(false);
    28  +           List<DirectiveNode> directives = ParseDirectives(false,
                    isQueryLocation: true);
29  29              SelectionSetNode selectionSet = ParseSelectionSet();
```

```
30  30            Location? location = CreateLocation(in start);
31  31
    @@ -139,7 +139,7 @@ private VariableDefinitionNode ParseVariableDefinition()
139 139                ? ParseValueLiteral(true)
140 140                : null;
141 141            List<DirectiveNode> directives =
142     -              ParseDirectives(true);
    142 +              ParseDirectives(true, isQueryLocation: true);
143 143
144 144            Location? location = CreateLocation(in start);
145 145
    @@ -181,6 +181,7 @@ private VariableNode ParseVariable()
181 181        [MethodImpl(MethodImplOptions.AggressiveInlining)]
182 182        private SelectionSetNode ParseSelectionSet()
183 183        {
    184 +          IncreaseDepth();
184 185            TokenInfo start = Start();
185 186
186 187            if (_reader.Kind != TokenKind.LeftBrace)
    @@ -208,6 +209,7 @@ private SelectionSetNode ParseSelectionSet()
208 209
209 210            Location? location = CreateLocation(in start);
210 211
    212 +          DecreaseDepth();
211 213            return new SelectionSetNode
212 214            (
213 215                location,
    @@ -253,7 +255,7 @@ private FieldNode ParseField()
253 255
254 256            List<ArgumentNode> arguments = ParseArguments(false);
255 257            INullabilityNode? required = ParseRequiredStatus();
256     -          List<DirectiveNode> directives = ParseDirectives(false);
    258 +          List<DirectiveNode> directives = ParseDirectives(false,
        isQueryLocation: true);
257 259            SelectionSetNode? selectionSet = _reader.Kind == TokenKind.LeftBrace
258 260                ? ParseSelectionSet()
```

```
259  261                 : null;
```

...rc/Language.Utf8/Utf8GraphQLParser.Types.cs

```
     @@ -12,6 +12,7 @@ public ref partial struct Utf8GraphQLParser
12   12          /// </summary>
13   13          private ITypeNode ParseTypeReference()
14   14          {
     15  +           IncreaseDepth();
15   16              ITypeNode type;
16   17              Location? location;
17   18
     @@ -40,6 +41,7 @@ private ITypeNode ParseTypeReference()
40   41                      MoveNext();
41   42                      location = CreateLocation(in start);
42   43
     44  +                   DecreaseDepth();
43   45                      return new NonNullTypeNode
44   46                      (
45   47                          location,
     @@ -50,6 +52,7 @@ private ITypeNode ParseTypeReference()
50   52                  Unexpected(TokenKind.Bang);
51   53              }
52   54
     55  +           DecreaseDepth();
53   56              return type;
54   57          }
55   58
```

...anguage.Utf8/Utf8GraphQLParser.Utilities.cs

```
     @@ -27,6 +27,25 @@ internal NameNode ParseName()
27   27          [MethodImpl(MethodImplOptions.AggressiveInlining)]
28   28          internal bool MoveNext() => _reader.MoveNext();
29   29
     30  +       [MethodImpl(MethodImplOptions.AggressiveInlining)]
     31  +       private void IncreaseDepth()
     32  +       {
```

```
      33  +         if (++_recursionDepth > _maxAllowedRecursionDepth)
      34  +         {
      35  +             throw new SyntaxException(
      36  +                 _reader,
      37  +                 string.Format(
      38  +                     Utf8GraphQLParser_Start_MaxAllowedRecursionDepthReached,
      39  +                     _maxAllowedRecursionDepth));
      40  +         }
      41  +     }
      42  +
      43  +     [MethodImpl(MethodImplOptions.AggressiveInlining)]
      44  +     private void DecreaseDepth()
      45  +     {
      46  +         --_recursionDepth;
      47  +     }
      48  +
30    49        [MethodImpl(MethodImplOptions.AggressiveInlining)]
31    50        private TokenInfo Start() =>
32    51            _createLocation
```

...c/Language.Utf8/Utf8GraphQLParser.Values.cs

```
          @@ -31,32 +31,37 @@ public ref partial struct Utf8GraphQLParser
31    31        /// </param>
32    32        internal IValueNode ParseValueLiteral(bool isConstant)
33    33        {
      34  +         IncreaseDepth();
      35  +
      36  +         IValueNode node;
      37  +
34    38            if (_reader.Kind == TokenKind.LeftBracket)
35    39            {
36        -             return ParseList(isConstant);
      40  +             node = ParseList(isConstant);
37    41            }
38        -
39        -         if (_reader.Kind == TokenKind.LeftBrace)
      42  +         else if (_reader.Kind == TokenKind.LeftBrace)
40    43            {
41        -             return ParseObject(isConstant);
```

```
      44 +              node = ParseObject(isConstant);
42    45                }
43       -
44       -          if (TokenHelper.IsScalarValue(in _reader))
      46 +          else if (TokenHelper.IsScalarValue(in _reader))
45    47                {
46       -              return ParseScalarValue();
      48 +              node = ParseScalarValue();
47    49                }
48       -
49       -          if (_reader.Kind == TokenKind.Name)
      50 +          else if (_reader.Kind == TokenKind.Name)
50    51                {
51       -              return ParseEnumValue();
      52 +              node = ParseEnumValue();
52    53                }
53       -
54       -          if (_reader.Kind == TokenKind.Dollar && !isConstant)
      54 +          else if (_reader.Kind == TokenKind.Dollar && !isConstant)
      55 +          {
      56 +              node = ParseVariable();
      57 +          }
      58 +          else
55    59                {
56       -              return ParseVariable();
      60 +              throw Unexpected(_reader.Kind);
57    61                }
58    62
59       -          throw Unexpected(_reader.Kind);
      63 +          DecreaseDepth();
      64 +          return node;
60    65        }
61    66
62    67        [MethodImpl(MethodImplOptions.AggressiveInlining)]
```

...uage/src/Language.Utf8/Utf8GraphQLParser.cs

```
@@ -9,8 +9,11 @@ public ref partial struct Utf8GraphQLParser
9     9     {
10    10        private readonly bool _createLocation;
```

```
11  11          private readonly bool _allowFragmentVars;
    12  +       private readonly int _maxAllowedDirectives;
    13  +       private readonly int _maxAllowedRecursionDepth;
12  14          private Utf8GraphQLReader _reader;
13  15          private StringValueNode? _description;
    16  +       private int _recursionDepth;
14  17
15  18          public Utf8GraphQLParser(
16  19              ReadOnlySpan<byte> graphQLData,
        @@ -24,6 +27,8 @@ public Utf8GraphQLParser(
24  27              options ??= ParserOptions.Default;
25  28              _createLocation = !options.NoLocations;
26  29              _allowFragmentVars = options.Experimental.AllowFragmentVariables;
    30  +           _maxAllowedDirectives = options.MaxAllowedDirectives;
    31  +           _maxAllowedRecursionDepth = options.MaxAllowedRecursionDepth;
27  32              _reader = new Utf8GraphQLReader(graphQLData);
28  33              _description = null;
29  34          }
        @@ -40,13 +45,16 @@ internal Utf8GraphQLParser(
40  45              options ??= ParserOptions.Default;
41  46              _createLocation = !options.NoLocations;
42  47              _allowFragmentVars = options.Experimental.AllowFragmentVariables;
    48  +           _maxAllowedDirectives = options.MaxAllowedDirectives;
    49  +           _maxAllowedRecursionDepth = options.MaxAllowedRecursionDepth;
43  50              _reader = reader;
44  51              _description = null;
45  52          }
46  53
47  54          public DocumentNode Parse()
48  55          {
49  56              var definitions = new List<IDefinitionNode>();
    57  +           _recursionDepth = 0;
50  58
51  59              TokenInfo start = Start();
52  60
```

Comments 0

Please sign in to comment.