ChilliCream / **graphql-platform** Public

Code Issues 287 Pull requests 89 Discussions Actions Projects

Commit b185eb2

michaelstaib authored last week · 91 / 94 · Verified

```
Add depth limit to GraphQL parser (#9530)
```

main-version-14 (#9530) · 14.3.1

1 parent 5b0cbe2 commit b185eb2

14 files changed +162 -24 lines changed

Top

Filter files…

- .build
  - Build.csproj
- src/HotChocolate
  - Core/src/Execution
    - DependencyInjection
      - RequestExecutorServiceCollectionExtensions.cs
    - Options
      - RequestParserOptions.cs
  - 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
- Primitives/test
  - Directory.Build.props

14 files changed +162 -24 lines changed

Search within code

.build/Build.csproj

```
@@ -2,7 +2,7 @@
2  2
3  3      <PropertyGroup>
4  4        <OutputType>Exe</OutputType>
5     -    <TargetFramework>net7.0</TargetFramework>
   5  +    <TargetFramework>net8.0</TargetFramework>
6  6        <RootNamespace></RootNamespace>
7  7        <NoWarn>CS0649;CS0169</NoWarn>
8  8        <NukeRootDirectory>..</NukeRootDirectory>
```

...questExecutorServiceCollectionExtensions.cs

```
@@ -96,9 +96,12 @@ public static IServiceCollection AddGraphQLCore(this
IServiceCollection services
96  96
97  97                  return new ParserOptions(
98  98                      noLocations: !options.IncludeLocations,
   99  +                   allowFragmentVariables: false,
99  100                     maxAllowedNodes: options.MaxAllowedNodes,
100 101                     maxAllowedTokens: options.MaxAllowedTokens,
101    -                   maxAllowedFields: options.MaxAllowedFields);
   102 +                   maxAllowedFields: options.MaxAllowedFields,
   103 +                   maxAllowedDirectives: options.MaxAllowedDirectives,
   104 +                   maxAllowedRecursionDepth:
    options.MaxAllowedRecursionDepth);
102 105             });
103 106
```

```
104  107            return services;
```

...c/Execution/Options/RequestParserOptions.cs

```
@@ -50,4 +50,17 @@ public sealed class RequestParserOptions
50  50         /// as fields is an easier way to estimate query size for GraphQL requests.
51  51         /// </summary>
52  52         public int MaxAllowedFields { get; set; } = 2048;
    53 +
    54 +     /// <summary>
    55 +     /// The maximum number of directives allowed per location (e.g. per field,
    56 +     /// per operation, per fragment definition). Repeatable directives can be
       used
    57 +     /// to exhaust CPU and memory resources if not limited.
    58 +     /// </summary>
    59 +     public int MaxAllowedDirectives { get; set; } = 4;
    60 +
    61 +     /// <summary>
    62 +     /// The maximum allowed recursion depth when parsing a document.
    63 +     /// This prevents stack overflow from deeply nested queries.
    64 +     /// </summary>
    65 +     public int MaxAllowedRecursionDepth { get; set; } = 200;
53  66   }
```

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

```
@@ -45,6 +45,50 @@ public ParserOptions(
45  45             MaxAllowedTokens = maxAllowedTokens;
46  46             MaxAllowedNodes = maxAllowedNodes;
47  47             MaxAllowedFields = maxAllowedFields;
    48 +         MaxAllowedDirectives = 4;
    49 +         MaxAllowedRecursionDepth = 200;
    50 +     }
    51 +
    52 +     /// <summary>
    53 +     /// Initializes a new instance of <see cref="ParserOptions"/>.
    54 +     /// </summary>
    55 +     /// <param name="noLocations">
    56 +     /// Defines that the parse shall not preserve syntax node locations.
    57 +     /// </param>
```

```
58  +      /// <param name="allowFragmentVariables">
59  +      /// Defines that the parser shall parse fragment variables.
60  +      /// </param>
61  +      /// <param name="maxAllowedNodes">
62  +      /// The maximum number of nodes allowed within a document.
63  +      /// </param>
64  +      /// <param name="maxAllowedTokens">
65  +      /// The maximum number of tokens allowed within a document.
66  +      /// </param>
67  +      /// <param name="maxAllowedFields">
68  +      /// The maximum number of fields allowed within a query document.
69  +      /// </param>
70  +      /// <param name="maxAllowedDirectives">
71  +      /// The maximum number of directives allowed per location.
72  +      /// </param>
73  +      /// <param name="maxAllowedRecursionDepth">
74  +      /// The maximum allowed recursion depth of a parsed document.
75  +      /// </param>
76  +      public ParserOptions(
77  +          bool noLocations,
78  +          bool allowFragmentVariables,
79  +          int maxAllowedNodes,
80  +          int maxAllowedTokens,
81  +          int maxAllowedFields,
82  +          int maxAllowedDirectives,
83  +          int maxAllowedRecursionDepth)
84  +      {
85  +          NoLocations = noLocations;
86  +          Experimental = new(allowFragmentVariables);
87  +          MaxAllowedTokens = maxAllowedTokens;
88  +          MaxAllowedNodes = maxAllowedNodes;
89  +          MaxAllowedFields = maxAllowedFields;
90  +          MaxAllowedDirectives = maxAllowedDirectives;
91  +          MaxAllowedRecursionDepth = maxAllowedRecursionDepth;
48  92         }
49  93
50  94         /// <summary>
    @@ -86,6 +130,18 @@ public ParserOptions(
86  130        /// </summary>
```

```
87  131         public int MaxAllowedFields { get; }
88  132
    133 +       /// <summary>
    134 +       /// The maximum number of directives allowed per location (e.g. per field,
    135 +       /// per operation, per fragment definition). Repeatable directives can be
        used
    136 +       /// to exhaust CPU and memory resources if not limited.
    137 +       /// </summary>
    138 +       public int MaxAllowedDirectives { get; }
    139 +
    140 +       /// <summary>
    141 +       /// Gets the maximum allowed recursion depth of a parsed document.
    142 +       /// </summary>
    143 +       public int MaxAllowedRecursionDepth { get; }
    144 +
89  145         /// <summary>
90  146         /// Gets the experimental parser options
91  147         /// which are by default switched of.
```

...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

```
    @@ -207,4 +207,10 @@
207 207       <data name="Utf8GraphQLParser_Start_MaxAllowedFieldsReached"
        xml:space="preserve">
208 208         <value>The GraphQL request document contains more than {0} fields. Parsing
        aborted.</value>
209 209       </data>
    210 +   <data name="Utf8GraphQLParser_ParseDirective_MaxAllowedDirectivesReached"
        xml:space="preserve">
    211 +     <value>A location in the GraphQL document contains more than {0}
        directives. Parsing aborted.</value>
    212 +   </data>
```

```
213 +   <data name="Utf8GraphQLParser_Start_MaxAllowedRecursionDepthReached"
      xml:space="preserve">
214 +     <value>Document exceeds the maximum allowed recursion depth of {0}. Parsing
      aborted.</value>
215 +   </data>
210 216   </root>
```

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

```
    @@ -1,4 +1,5 @@
1   1   using System.Runtime.CompilerServices;
    2 + using static HotChocolate.Language.Properties.LangUtf8Resources;
2   3
3   4   namespace HotChocolate.Language;
4   5
    @@ -64,7 +65,7 @@ private NameNode ParseDirectiveLocation()
64  65          throw Unexpected(kind);
65  66      }
66  67
67    - private List<DirectiveNode> ParseDirectives(bool isConstant)
    68 + private List<DirectiveNode> ParseDirectives(bool isConstant, bool
      isQueryLocation = false)
68  69      {
69  70          if (_reader.Kind == TokenKind.At)
70  71          {
    @@ -73,6 +74,15 @@ private List<DirectiveNode> ParseDirectives(bool
      isConstant)
73  74              while (_reader.Kind == TokenKind.At)
74  75              {
75  76                  list.Add(ParseDirective(isConstant));
    77 +
    78 +                if (isQueryLocation && list.Count > _maxAllowedDirectives)
    79 +                {
    80 +                    throw new SyntaxException(
    81 +                        _reader,
    82 +                        string.Format(
    83 +
      Utf8GraphQLParser_ParseDirective_MaxAllowedDirectivesReached,
    84 +                            _maxAllowedDirectives));
    85 +                }
```

```
76  86                }
77  87
78  88                return list;
```

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

```
    @@ -53,7 +53,7 @@ private FragmentDefinitionNode ParseFragmentDefinition()
53  53                  ParseVariableDefinitions();
54  54                ExpectOnKeyword();
55  55                var typeCondition = ParseNamedType();
56      -             var directives = ParseDirectives(false);
    56  +             var directives = ParseDirectives(false, isQueryLocation: true);
57  57                var selectionSet = ParseSelectionSet();
58  58                var location = CreateLocation(in start);
59  59
    @@ -72,7 +72,7 @@ private FragmentDefinitionNode ParseFragmentDefinition()
72  72                var name = ParseFragmentName();
73  73                ExpectOnKeyword();
74  74                var typeCondition = ParseNamedType();
75      -             var directives = ParseDirectives(false);
    75  +             var directives = ParseDirectives(false, isQueryLocation: true);
76  76                var selectionSet = ParseSelectionSet();
77  77                var location = CreateLocation(in start);
78  78
    @@ -99,7 +99,7 @@ private FragmentSpreadNode ParseFragmentSpread(in TokenInfo start)
99  99          private FragmentSpreadNode ParseFragmentSpread(in TokenInfo start)
100 100         {
101 101             var name = ParseFragmentName();
102     -         var directives = ParseDirectives(false);
    102 +         var directives = ParseDirectives(false, isQueryLocation: true);
103 103             var location = CreateLocation(in start);
104 104
105 105             return new FragmentSpreadNode
    @@ -125,7 +125,7 @@ private InlineFragmentNode ParseInlineFragment(
125 125             in TokenInfo start,
126 126             NamedTypeNode? typeCondition)
127 127         {
128     -         var directives = ParseDirectives(false);
    128 +         var directives = ParseDirectives(false, isQueryLocation: true);
129 129             var selectionSet = ParseSelectionSet();
```

```
130  130            var location = CreateLocation(in start);
131  131
```

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

```
@@ -21,7 +21,7 @@ private OperationDefinitionNode ParseOperationDefinition()
21   21             var operation = ParseOperationType();
22   22             var name = _reader.Kind == TokenKind.Name ? ParseName() : null;
23   23             var variableDefinitions = ParseVariableDefinitions();
24      -          var directives = ParseDirectives(false);
     24 +          var directives = ParseDirectives(false, isQueryLocation: true);
25   25             var selectionSet = ParseSelectionSet();
26   26             var location = CreateLocation(in start);
27   27
@@ -127,7 +127,7 @@ private VariableDefinitionNode ParseVariableDefinition()
127  127                ? ParseValueLiteral(true)
128  128                : null;
129  129            var directives =
130     -              ParseDirectives(isConstant: true);
     130 +              ParseDirectives(isConstant: true, isQueryLocation: true);
131  131
132  132            var location = CreateLocation(in start);
133  133
@@ -163,6 +163,7 @@ private VariableNode ParseVariable()
163  163        /// </summary>
164  164        private SelectionSetNode ParseSelectionSet()
165  165        {
     166 +          IncreaseDepth();
166  167            var start = Start();
167  168
168  169            if (_reader.Kind != TokenKind.LeftBrace)
@@ -191,6 +192,7 @@ private SelectionSetNode ParseSelectionSet()
191  192
192  193            var location = CreateLocation(in start);
193  194
     195 +          DecreaseDepth();
194  196            return new SelectionSetNode(
```

```
195  197                location,
196  198                selections);
     @@ -240,7 +242,7 @@ private FieldNode ParseField()
240  242            }
241  243
242  244            var arguments = ParseArguments(false);
243       -        var directives = ParseDirectives(false);
          245 +    var directives = ParseDirectives(false, isQueryLocation: true);
244  246            var selectionSet = _reader.Kind == TokenKind.LeftBrace
245  247                ? ParseSelectionSet()
246  248                : 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

```
@@ -24,6 +24,25 @@ private NameNode ParseName()
24  24      [MethodImpl(MethodImplOptions.AggressiveInlining)]
25  25      private bool MoveNext() => _reader.MoveNext();
26  26
    27  +   [MethodImpl(MethodImplOptions.AggressiveInlining)]
    28  +   private void IncreaseDepth()
    29  +   {
    30  +       if (++_recursionDepth > _maxAllowedRecursionDepth)
    31  +       {
    32  +           throw new SyntaxException(
    33  +               _reader,
    34  +               string.Format(
    35  +                   Utf8GraphQLParser_Start_MaxAllowedRecursionDepthReached,
    36  +                   _maxAllowedRecursionDepth));
    37  +       }
    38  +   }
    39  +
    40  +   [MethodImpl(MethodImplOptions.AggressiveInlining)]
    41  +   private void DecreaseDepth()
    42  +   {
    43  +       --_recursionDepth;
    44  +   }
    45  +
27  46      [MethodImpl(MethodImplOptions.AggressiveInlining)]
28  47      private TokenInfo Start()
29  48      {
```

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

```
@@ -29,32 +29,37 @@ public ref partial struct Utf8GraphQLParser
29  29      /// </param>
30  30      private IValueNode ParseValueLiteral(bool isConstant)
31  31      {
    32  +       IncreaseDepth();
    33  +
    34  +       IValueNode node;
```

```
      35 +
32    36             if (_reader.Kind == TokenKind.LeftBracket)
33    37             {
34       -               return ParseList(isConstant);
      38 +               node = ParseList(isConstant);
35    39             }
36       -
37       -           if (_reader.Kind == TokenKind.LeftBrace)
      40 +           else if (_reader.Kind == TokenKind.LeftBrace)
38    41             {
39       -               return ParseObject(isConstant);
      42 +               node = ParseObject(isConstant);
40    43             }
41       -
42       -           if (TokenHelper.IsScalarValue(ref _reader))
      44 +           else if (TokenHelper.IsScalarValue(ref _reader))
43    45             {
44       -               return ParseScalarValue();
      46 +               node = ParseScalarValue();
45    47             }
46       -
47       -           if (_reader.Kind == TokenKind.Name)
      48 +           else if (_reader.Kind == TokenKind.Name)
48    49             {
49       -               return ParseEnumValue();
      50 +               node = ParseEnumValue();
50    51             }
51       -
52       -           if (_reader.Kind == TokenKind.Dollar && !isConstant)
      52 +           else if (_reader.Kind == TokenKind.Dollar && !isConstant)
      53 +           {
      54 +               node = ParseVariable();
      55 +           }
      56 +           else
53    57             {
54       -               return ParseVariable();
      58 +               throw Unexpected(_reader.Kind);
55    59             }
56    60
57       -           throw Unexpected(_reader.Kind);
```

```
      61 +          DecreaseDepth();
      62 +          return node;
58    63          }
59    64
60    65          [MethodImpl(MethodImplOptions.AggressiveInlining)]
```

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

```
@@ -10,10 +10,13 @@ public ref partial struct Utf8GraphQLParser
10    10          private readonly bool _allowFragmentVars;
11    11          private readonly int _maxAllowedNodes;
12    12          private readonly int _maxAllowedFields;
      13 +        private readonly int _maxAllowedDirectives;
      14 +        private readonly int _maxAllowedRecursionDepth;
13    15          private Utf8GraphQLReader _reader;
14    16          private StringValueNode? _description;
15    17          private int _parsedNodes;
16    18          private int _parsedFields;
      19 +        private int _recursionDepth;
17    20
18    21          public Utf8GraphQLParser(
19    22              ReadOnlySpan<byte> graphQLData,
@@ -29,6 +32,8 @@ public Utf8GraphQLParser(
29    32              _allowFragmentVars = options.Experimental.AllowFragmentVariables;
30    33              _maxAllowedNodes = options.MaxAllowedNodes;
31    34              _maxAllowedFields = options.MaxAllowedFields;
      35 +            _maxAllowedDirectives = options.MaxAllowedDirectives;
      36 +            _maxAllowedRecursionDepth = options.MaxAllowedRecursionDepth;
32    37              _reader = new Utf8GraphQLReader(graphQLData, options.MaxAllowedTokens);
33    38              _description = null;
34    39          }
@@ -47,6 +52,8 @@ internal Utf8GraphQLParser(
47    52              _allowFragmentVars = options.Experimental.AllowFragmentVariables;
48    53              _maxAllowedNodes = options.MaxAllowedNodes;
49    54              _maxAllowedFields = options.MaxAllowedFields;
      55 +            _maxAllowedDirectives = options.MaxAllowedDirectives;
      56 +            _maxAllowedRecursionDepth = options.MaxAllowedRecursionDepth;
50    57              _reader = reader;
51    58              _description = null;
52    59          }
```

```
@@ -64,6 +71,7 @@ internal Utf8GraphQLParser(
64  71          public DocumentNode Parse()
65  72          {
66  73              _parsedNodes = 0;
    74  +           _recursionDepth = 0;
67  75              var definitions = new List<IDefinitionNode>();
68  76
69  77              var start = Start();
```

...olate/Primitives/test/Directory.Build.props

```
@@ -2,6 +2,7 @@
2   2     <Import Project="$([MSBuild]::GetPathOfFileAbove('Directory.Build.props',
          '$(MSBuildThisFileDirectory)..\'))" />
3   3
4   4     <PropertyGroup>
    5   +   <TargetFrameworks>$(TestTargetFrameworks)</TargetFrameworks>
5   6       <IsPackable>false</IsPackable>
6   7       <GenerateDocumentationFile>false</GenerateDocumentationFile>
7   8     </PropertyGroup>
```

Comments 0

Please sign in to comment.