ChilliCream / **graphql-platform** Public

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

Commit `08c0caa`

michaelstaib authored last week · ✕ 0 / 1 · Verified

```
Add depth limit to GraphQL parser (#9531)
```

`main-version-13` (EvilVir/graphql-platform#1, #9531) · 13.9.16

1 parent `e1b4824` commit `08c0caa`

12 files changed +139 -23 lines changed

Top

Filter files…

- 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

12 files changed +139 -23 lines changed

Search within code

...questExecutorServiceCollectionExtensions.cs

```
@@ -91,9 +91,12 @@ public static IServiceCollection AddGraphQLCore(this
IServiceCollection services
91  91
92  92                  return new ParserOptions(
93  93                      noLocations: !options.IncludeLocations,
    94  +                   allowFragmentVariables: false,
94  95                      maxAllowedNodes: options.MaxAllowedNodes,
95  96                      maxAllowedTokens: options.MaxAllowedTokens,
96      -                   maxAllowedFields: options.MaxAllowedFields);
    97  +                   maxAllowedFields: options.MaxAllowedFields,
    98  +                   maxAllowedDirectives: options.MaxAllowedDirectives,
    99  +                   maxAllowedRecursionDepth:
    options.MaxAllowedRecursionDepth);
97  100             });
98  101
99  102         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

```
@@ -42,6 +42,29 @@ public ParserOptions(
42    42             MaxAllowedTokens = maxAllowedTokens;
43    43             MaxAllowedNodes = maxAllowedNodes;
44    44             MaxAllowedFields = maxAllowedFields;
      45 +         MaxAllowedDirectives = 4;
      46 +         MaxAllowedRecursionDepth = 200;
      47 +     }
      48 +
      49 +     /// <summary>
      50 +     /// Initializes a new instance of <see cref="ParserOptions"/> with security
         limits.
      51 +     /// </summary>
      52 +     public ParserOptions(
      53 +         bool noLocations,
      54 +         bool allowFragmentVariables,
      55 +         int maxAllowedNodes,
      56 +         int maxAllowedTokens,
      57 +         int maxAllowedFields,
      58 +         int maxAllowedDirectives,
      59 +         int maxAllowedRecursionDepth)
      60 +     {
      61 +         NoLocations = noLocations;
      62 +         Experimental = new(allowFragmentVariables);
      63 +         MaxAllowedTokens = maxAllowedTokens;
      64 +         MaxAllowedNodes = maxAllowedNodes;
      65 +         MaxAllowedFields = maxAllowedFields;
      66 +         MaxAllowedDirectives = maxAllowedDirectives;
      67 +         MaxAllowedRecursionDepth = maxAllowedRecursionDepth;
45    68         }
46    69
47    70         /// <summary>
```

```
@@ -83,6 +106,18 @@ public ParserOptions(
83  106         /// </summary>
84  107         public int MaxAllowedFields { get; }
85  108
    109 +     /// <summary>
    110 +     /// The maximum number of directives allowed per location (e.g. per field,
    111 +     /// per operation, per fragment definition). Repeatable directives can be used
    112 +     /// to exhaust CPU and memory resources if not limited.
    113 +     /// </summary>
    114 +     public int MaxAllowedDirectives { get; }
    115 +
    116 +     /// <summary>
    117 +     /// Gets the maximum allowed recursion depth of a parsed document.
    118 +     /// </summary>
    119 +     public int MaxAllowedRecursionDepth { get; }
    120 +
86  121         /// <summary>
87  122         /// Gets the experimental parser options
88  123         /// 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

```
@@ -204,4 +204,10 @@
204 204   <data name="Utf8GraphQLParser_Start_MaxAllowedFieldsReached" xml:space="preserve">
205 205     <value>The GraphQL request document contains more than {0} fields. Parsing aborted.</value>
206 206   </data>
    207 +   <data name="Utf8GraphQLParser_ParseDirective_MaxAllowedDirectivesReached" xml:space="preserve">
```

```
208 +     <value>A location in the GraphQL document contains more than {0} directives. Parsing aborted.</value>
209 +   </data>
210 +   <data name="Utf8GraphQLParser_Start_MaxAllowedRecursionDepthReached" xml:space="preserve">
211 +     <value>Document exceeds the maximum allowed recursion depth of {0}. Parsing aborted.</value>
212 +   </data>
207 213   </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
    @@ -65,7 +66,7 @@ private NameNode ParseDirectiveLocation()
65  66              throw Unexpected(kind);
66  67          }
67  68
68      -     private List<DirectiveNode> ParseDirectives(bool isConstant)
    69  +     private List<DirectiveNode> ParseDirectives(bool isConstant, bool isQueryLocation = false)
69  70          {
70  71              if (_reader.Kind == TokenKind.At)
71  72              {
    @@ -74,6 +75,15 @@ private List<DirectiveNode> ParseDirectives(bool isConstant)
74  75                  while (_reader.Kind == TokenKind.At)
75  76                  {
76  77                      list.Add(ParseDirective(isConstant));
    78  +
    79  +                 if (isQueryLocation && list.Count > _maxAllowedDirectives)
    80  +                 {
    81  +                     throw new SyntaxException(
    82  +                         _reader,
    83  +                         string.Format(
```

```
      84 +
         Utf8GraphQLParser_ParseDirective_MaxAllowedDirectivesReached,
      85 +                             _maxAllowedDirectives));
      86 +                 }
77    87                 }
78    88
79    89                 return list;
```

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

```
@@ -55,7 +55,7 @@ private FragmentDefinitionNode ParseFragmentDefinition()
55    55                   ParseVariableDefinitions();
56    56                 ExpectOnKeyword();
57    57                 var typeCondition = ParseNamedType();
58      -                var directives = ParseDirectives(false);
      58 +                var directives = ParseDirectives(false, isQueryLocation: true);
59    59                 var selectionSet = ParseSelectionSet();
60    60                 var location = CreateLocation(in start);
61    61
@@ -74,7 +74,7 @@ private FragmentDefinitionNode ParseFragmentDefinition()
74    74                 var name = ParseFragmentName();
75    75                 ExpectOnKeyword();
76    76                 var typeCondition = ParseNamedType();
77      -                var directives = ParseDirectives(false);
      77 +                var directives = ParseDirectives(false, isQueryLocation: true);
78    78                 var selectionSet = ParseSelectionSet();
79    79                 var location = CreateLocation(in start);
80    80
@@ -101,7 +101,7 @@ private FragmentDefinitionNode ParseFragmentDefinition()
101   101        private FragmentSpreadNode ParseFragmentSpread(in TokenInfo start)
102   102        {
103   103            var name = ParseFragmentName();
104     -          var directives = ParseDirectives(false);
      104 +          var directives = ParseDirectives(false, isQueryLocation: true);
105   105            var location = CreateLocation(in start);
106   106
107   107            return new FragmentSpreadNode
@@ -127,7 +127,7 @@ private InlineFragmentNode ParseInlineFragment(
127   127            in TokenInfo start,
128   128            NamedTypeNode? typeCondition)
```

```
129  129        {
130       -          var directives = ParseDirectives(false);
     130  +          var directives = ParseDirectives(false, isQueryLocation: true);
131  131           var selectionSet = ParseSelectionSet();
132  132           var location = CreateLocation(in start);
133  133
```

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

```
@@ -23,7 +23,7 @@ private OperationDefinitionNode ParseOperationDefinition()
23   23           var operation = ParseOperationType();
24   24           var name = _reader.Kind == TokenKind.Name ? ParseName() : null;
25   25           var variableDefinitions = ParseVariableDefinitions();
26        -          var directives = ParseDirectives(false);
     26   +          var directives = ParseDirectives(false, isQueryLocation: true);
27   27           var selectionSet = ParseSelectionSet();
28   28           var location = CreateLocation(in start);
29   29
@@ -133,7 +133,7 @@ private VariableDefinitionNode ParseVariableDefinition()
133  133               ? ParseValueLiteral(true)
134  134               : null;
135  135           var directives =
136       -              ParseDirectives(true);
     136  +              ParseDirectives(isConstant: true, isQueryLocation: true);
137  137
138  138           var location = CreateLocation(in start);
139  139
@@ -173,6 +173,7 @@ private VariableNode ParseVariable()
173  173       /// </summary>
174  174       private SelectionSetNode ParseSelectionSet()
175  175       {
     176  +          IncreaseDepth();
176  177           var start = Start();
177  178
178  179           if (_reader.Kind != TokenKind.LeftBrace)
@@ -200,6 +201,7 @@ private SelectionSetNode ParseSelectionSet()
200  201
```

```
201 202             var location = CreateLocation(in start);
202 203
    204 +           DecreaseDepth();
203 205             return new SelectionSetNode
204 206             (
205 207                 location,
    @@ -252,7 +254,7 @@ private FieldNode ParseField()
252 254
253 255             var arguments = ParseArguments(false);
254 256             var required = ParseRequiredStatus();
255     -           var directives = ParseDirectives(false);
    257 +           var directives = ParseDirectives(false, isQueryLocation: true);
256 258             var selectionSet = _reader.Kind == TokenKind.LeftBrace
257 259                 ? ParseSelectionSet()
258 260                 : 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        {
```

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

```
@@ -32,32 +32,37 @@ public ref partial struct Utf8GraphQLParser
32    32        /// </param>
33    33        internal IValueNode ParseValueLiteral(bool isConstant)
```

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

```
      61 +              throw Unexpected(_reader.Kind);
58    62              }
59    63
60        -         throw Unexpected(_reader.Kind);
      64 +         DecreaseDepth();
      65 +         return node;
61    66          }
62    67
63    68          [MethodImpl(MethodImplOptions.AggressiveInlining)]
```

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

```
      @@ -12,10 +12,13 @@ public ref partial struct Utf8GraphQLParser
12    12          private readonly bool _allowFragmentVars;
13    13          private readonly int _maxAllowedNodes;
14    14          private readonly int _maxAllowedFields;
      15 +        private readonly int _maxAllowedDirectives;
      16 +        private readonly int _maxAllowedRecursionDepth;
15    17          private Utf8GraphQLReader _reader;
16    18          private StringValueNode? _description;
17    19          private int _parsedNodes;
18    20          private int _parsedFields;
      21 +        private int _recursionDepth;
19    22
20    23          public Utf8GraphQLParser(
21    24              ReadOnlySpan<byte> graphQLData,
      @@ -31,6 +34,8 @@ public Utf8GraphQLParser(
31    34              _allowFragmentVars = options.Experimental.AllowFragmentVariables;
32    35              _maxAllowedNodes = options.MaxAllowedNodes;
33    36              _maxAllowedFields = options.MaxAllowedFields;
      37 +            _maxAllowedDirectives = options.MaxAllowedDirectives;
      38 +            _maxAllowedRecursionDepth = options.MaxAllowedRecursionDepth;
34    39              _reader = new Utf8GraphQLReader(graphQLData, options.MaxAllowedTokens);
35    40              _description = null;
36    41          }
      @@ -49,6 +54,8 @@ internal Utf8GraphQLParser(
49    54              _allowFragmentVars = options.Experimental.AllowFragmentVariables;
50    55              _maxAllowedNodes = options.MaxAllowedNodes;
51    56              _maxAllowedFields = options.MaxAllowedFields;
      57 +            _maxAllowedDirectives = options.MaxAllowedDirectives;
```

```
      58  +         _maxAllowedRecursionDepth = options.MaxAllowedRecursionDepth;
52    59            _reader = reader;
53    60            _description = null;
54    61        }
          @@ -72,6 +79,7 @@ internal Utf8GraphQLParser(
72    79        public DocumentNode Parse()
73    80        {
74    81            _parsedNodes = 0;
      82  +         _recursionDepth = 0;
75    83            var definitions = new List<IDefinitionNode>();
76    84
77    85            var start = Start();
```

Comments 0

Please sign in to comment.