

LinksPlatform's Platform.Data.Doubles.Gql Class Library

1.1 ./csharp/Platform.Data.Doubles.Gql.Tests/MutationTests.cs

```
1  using GraphQL;
2  using GraphQL.SystemTextJson;
3  using Newtonsoft.Json;
4  using Newtonsoft.Json.Linq;
5  using Platform.Data.Doubles.Gql.Schema;
6  using Platform.Data.Doubles.Memory;
7  using Platform.Data.Doubles.Memory.United.Generic;
8  using Platform.IO;
9  using Platform.Memory;
10 using System;
11 using System.Collections.Generic;
12 using System.Linq;
13 using Xunit;
14 using TLinkAddress = System.UInt64;
15
16 namespace Platform.Data.Doubles.Gql.Tests
17 {
18     public class MutationTests
19     {
20         public static EqualityComparer<TLinkAddress> EqualityComparer =
21             ↪ EqualityComparer<TLinkAddress>.Default;
22         public static ILinks<ulong> CreateLinks() => CreateLinks<ulong>(new TemporaryFile());
23         public static ILinks<TLinkAddress> CreateLinks<TLinkAddress>(string dataDBFilename)
24         {
25             var linksConstants = new LinksConstants<TLinkAddress>(true);
26             return new UnitedMemoryLinks<TLinkAddress>(new
27                 ↪ FileMappedResizableDirectMemory(dataDBFilename),
28                 ↪ UnitedMemoryLinks<TLinkAddress>.DefaultLinksSizeStep, linksConstants,
29                 ↪ IndexTreeType.Default);
30         }
31
32         [Fact]
33         public void InsertLinksOne()
34         {
35             var links = CreateLinks();
36             LinksSchema linksSchema = new(links, new DefaultServiceProvider());
37             var jsonTask = linksSchema.ExecuteAsync(_ => { _.Query = @"
38 mutation {
39     insert_links_one(object: {from_id: 1, to_id: 1}) {
40         id
41         from_id
42         to_id
43     }
44 }"; });
45             dynamic result =
46                 ↪ Newtonsoft.Json.JsonConvert.DeserializeObject<dynamic>(jsonTask.Result);
47             if (result.ContainsKey("errors"))
48             {
49                 throw new Exception(result.errors.ToString());
50             }
51         }
52
53         [Fact]
54         public void InsertLinks()
55         {
56             var links = CreateLinks();
57             LinksSchema linksSchema = new(links, new DefaultServiceProvider());
58             var jsonTask = linksSchema.ExecuteAsync(_ => { _.Query = @"
59 mutation {
60     insert_links(objects: [{ from_id: 1, to_id: 1 }, { from_id: 2, to_id: 2 }]) {
61         returning {
62             id
63             from_id
64             to_id
65         }
66     }
67 }"; });
68             dynamic result =
69                 ↪ Newtonsoft.Json.JsonConvert.DeserializeObject<dynamic>(jsonTask.Result);
70             if (result.ContainsKey("errors"))
71             {
72                 throw new Exception(result.errors.ToString());
73             }
74         }
75     }
76 }
```

```

73 [Fact]
74 public void UpdateLinks()
75 {
76     var links = CreateLinks();
77     LinksSchema linksSchema = new(links, new DefaultServiceProvider());
78     var jsonTask = linksSchema.ExecuteAsync(_ => { _.Query = @"
79     mutation {
80         ↪ update_links(_set: { from_id: 1, to_id: 2 }, where: { from_id: { _eq: 2 }, to_id: {
81             ↪ _eq: 2 } }) {
82             ↪     returning {
83                 ↪     id
84                 ↪     from_id
85                 ↪     to_id
86             ↪     }
87         ↪ }
88     "; });
89     dynamic result =
90     ↪     Newtonsoft.Json.JsonConvert.DeserializeObject<dynamic>(jsonTask.Result);
91     if (result.ContainsKey("errors"))
92     {
93         throw new Exception(result.errors.ToString());
94     }
95 }
96 [Fact]
97 public void DeleteLinks()
98 {
99
100     var links = CreateLinks();
101     LinksSchema linksSchema = new(links, new DefaultServiceProvider());
102     var jsonTask = linksSchema.ExecuteAsync(_ => { _.Query = @"
103     mutation {
104         delete_links(where: { from_id: { _eq: 1 }, to_id: { _eq: 1 } }) {
105             ↪     returning {
106                 ↪     id
107                 ↪     from_id
108                 ↪     to_id
109             ↪     }
110         ↪ }
111     "; });
112     dynamic result =
113     ↪     Newtonsoft.Json.JsonConvert.DeserializeObject<dynamic>(jsonTask.Result);
114     if (result.ContainsKey("errors"))
115     {
116         throw new Exception(result.errors.ToString());
117     }
118 }
119 [Fact]
120 public void CreateZeroZeroAndUpdateToOneOneById()
121 {
122     var links = CreateLinks();
123     LinksSchema linksSchema = new(links, new DefaultServiceProvider());
124     var jsonTask = linksSchema.ExecuteAsync(_ => { _.Query = @"
125     mutation {
126         insert_links_one(object: {from_id: 0, to_id: 0}) {
127             ↪     id
128             ↪     from_id
129             ↪     to_id
130         ↪     }
131     }
132     "; });
133     var jsonSerializer = new JsonSerializer();
134     var jsonResponse = jsonTask.Result;
135     Assert.False(JObject.Parse(jsonResponse).ContainsKey("errors"));
136     jsonTask = linksSchema.ExecuteAsync(_ => { _.Query = @"
137     mutation {
138         update_links(_set: { from_id: 1, to_id: 1 }, where: { id: { _eq: 1 } }) {
139             ↪     returning {
140                 ↪     id
141                 ↪     from_id
142                 ↪     to_id
143             ↪     }
144         ↪ }
145     }
146     "; });
147 }

```

```

148         dynamic result =
149             ↪ Newtonsoft.Json.JsonConvert.DeserializeObject<dynamic>(jsonTask.Result);
150         if (result.ContainsKey("errors"))
151         {
152             throw new Exception(result.errors.ToString());
153         }
154         Assert.True(1 == Convert.ToInt32(result.data.update_links.returning[0].id));
155     }
156 }

```

1.2 ./csharp/Platform.Data.Doublets.Gql.Tests/QueryTest.cs

```

1  using GraphQL;
2  using GraphQL.SystemTextJson;
3  using Newtonsoft.Json.Linq;
4  using Platform.Data.Doublets.Gql.Schema;
5  using Platform.Data.Doublets.Memory;
6  using Platform.Data.Doublets.Memory.United.Generic;
7  using Platform.IO;
8  using Platform.Memory;
9  using Xunit;
10 using TLinkAddress = System.UInt64;
11
12 namespace Platform.Data.Doublets.Gql.Tests
13 {
14     public class QueryTests
15     {
16         public static ILinks<ulong> CreateLinks() => CreateLinks<ulong>(new TemporaryFile());
17
18         public static ILinks<TLinkAddress> CreateLinks<TLinkAddress>(string dataDbFilename)
19         {
20             var linksConstants = new LinksConstants<TLinkAddress>(true);
21             return new UnitedMemoryLinks<TLinkAddress>(new
22                 ↪ FileMappedResizableDirectMemory(dataDbFilename),
23                 ↪ UnitedMemoryLinks<TLinkAddress>.DefaultLinksSizeStep, linksConstants,
24                 ↪ IndexTreeType.Default);
25         }
26
27         [InlineData(@"
28 {
29     links {
30         id
31     }
32 }
33 ")
34 ]
35 [InlineData(@"
36 {
37     links(
38         where: { id: { _eq: 1 }, from_id: { _eq: 1 }, to_id: { _eq: 1 } }
39         distinct_on: [from_id]
40         order_by: { id: asc }
41         offset: 0
42         limit: 1
43     ) {
44         id
45         from_id
46         from {
47             id
48             from_id
49             to_id
50         }
51         out {
52             id
53             from_id
54             to_id
55         }
56         to_id
57         to {
58             id
59             from_id
60             to_id
61         }
62         in {
63             id
64             from_id
65             to_id
66         }
67     }
68 }
69 ")
70 ]

```

```

66 [InlineData(@"
67 {
68     links(
69         where: { id: { _eq: 1 }, from_id: { _eq: 1 }, to_id: { _eq: 1 } }
70         distinct_on: [from_id]
71         order_by: { id: asc }
72         offset: 0
73         limit: 1
74     ) {
75         id
76         from_id
77         from {
78             id
79             from_id
80             to_id
81         }
82         out(
83             where: { from_id: { _eq: 1 }, to_id: { _eq: 1 } }
84             distinct_on: [from_id]
85             order_by: { id: asc }
86             offset: 0
87             limit: 1
88         ) {
89             id
90             from_id
91             to_id
92         }
93         to_id
94         to {
95             id
96             from_id
97             to_id
98         }
99         in(
100             where: { from_id: { _eq: 1 }, to_id: { _eq: 1 } }
101             distinct_on: [from_id]
102             order_by: { id: asc }
103             offset: 0
104             limit: 1
105         ) {
106             id
107             from_id
108             to_id
109         }
110     }
111 }
112 ")])
113 [Theory]
114 public void QueryData(string query)
115 {
116     var links = CreateLinks();
117     LinksSchema linksSchema = new(links, new DefaultServiceProvider());
118     var jsonTask = linksSchema.ExecuteAsync(_ => { _.Query = query; });
119     var response = JObject.Parse(jsonTask.Result);
120     var error = response.ContainsKey("errors");
121     Assert.False(error);
122 }
123 }
124 }

```

1.3 ./csharp/Platform.Data.Doublets.Gql.Tests/TestExtensions.cs

```

1 using System;
2 using System.Diagnostics;
3 using System.IO;
4 using System.Threading;
5
6 namespace Platform.Data.Doublets.Gql.Tests;
7
8 public static class TestExtensions
9 {
10     public static Process RunServer(string tempFilePath)
11     {
12         var currentAssemblyDirectory = Directory.GetCurrentDirectory();
13         var currentProjectDirectory = Path.GetFullPath(Path.Combine(currentAssemblyDirectory,
14             ↪ "..", "..", ".."));
15         var serverProjectDirectory = Path.GetFullPath(Path.Combine(currentProjectDirectory,
16             ↪ "..", "Platform.Data.Doublets.Gql.Server"));
17         var processStartInfo = new ProcessStartInfo { WorkingDirectory = serverProjectDirectory,
18             ↪ FileName = "dotnet", Arguments = $"run -f net5 {tempFilePath}",
19             ↪ RedirectStandardOutput = true, RedirectStandardInput = true};

```

```

16     var process = Process.Start(processStartInfo);
17     if (null == process || process.HasExited)
18     {
19         throw new Exception("Failed to start server process");
20     }
21     return process;
22 }
23
24 public static Uri GetEndPointFromServerProcess(Process process)
25 {
26     while (true)
27     {
28         var standartOutput = process?.StandardOutput;
29         if(standartOutput == null)
30         {
31             Thread.Sleep(TimeSpan.FromSeconds(1));
32             continue;
33         }
34         var processOutputLine = standartOutput.ReadLine();
35         if (string.IsNullOrEmpty(processOutputLine))
36         {
37             Thread.Sleep(TimeSpan.FromSeconds(1));
38             continue;
39         }
40         if (processOutputLine.Contains("Unable to start"))
41         {
42             throw new Exception("Unable to start.");
43         }
44         if (processOutputLine.Contains("Now listening on: "))
45         {
46             var index = processOutputLine.IndexOf("Now listening on: ",
47                 ↳ StringComparison.Ordinal) + "Now listening on: ".Length;
48             var uriString = processOutputLine.Substring(index);
49             return new Uri($"{uriString}/v1/graphql");
50         }
51     }
52 }

```

Index

./csharp/Platform.Data.Doublets.Gql.Tests/MutationTests.cs, 1

./csharp/Platform.Data.Doublets.Gql.Tests/QueryTest.cs, 3

./csharp/Platform.Data.Doublets.Gql.Tests/TestExtensions.cs, 4