

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

1.1 ./csharp/Platform.Data.Doublets.Gql.Tests/ClientTests.cs

```
1  using GraphQL.Client.Http;
2  using GraphQL.Client.Serializer.Newtonsoft;
3  using Microsoft.AspNetCore.Hosting;
4  using Microsoft.Extensions.Hosting;
5  using Platform.Data.Doublets.Gql.Client;
6  using Platform.Data.Doublets.Gql.Server;
7  using Platform.Data.Doublets.Memory;
8  using Platform.Data.Doublets.Memory.United.Generic;
9  using Platform.Memory;
10 using Serilog;
11 using System;
12 using System.Collections.Generic;
13 using System.Diagnostics;
14 using System.IO;
15 using System.Reflection;
16 using System.Threading;
17 using Xunit;
18 using TLinkAddress = System.UInt64;
19
20 namespace Platform.Data.Doublets.Gql.Tests;
21
22 public class ClientTests : IDisposable
23 {
24     private readonly LinksConstants<TLinkAddress> _constants;
25     private readonly LinksGqlAdapter _linksGqlAdapter;
26     private readonly Process _serverProcess;
27     private readonly Uri _endPoint;
28     public string TempFilePath = new IO.TemporaryFile();
29
30     public ClientTests()
31     {
32         _constants = new LinksConstants<TLinkAddress>(true);
33         _serverProcess = TestExtensions.RunServer(TempFilePath);
34         _endPoint = TestExtensions.GetEndPointFromServerProcess(_serverProcess);
35         var graphQlClient = new GraphQLHttpClient(_endPoint, new Newtonsoft.Json.Serializer());
36         _linksGqlAdapter = new LinksGqlAdapter(graphQlClient, _constants);
37     }
38
39     public void Dispose()
40     {
41         _serverProcess.Kill(true);
42     }
43
44
45
46     private void TestCud()
47     {
48         TLinkAddress linksAmount = 5;
49         // Create
50         for (TLinkAddress i = 1; i <= linksAmount; i++)
51         {
52             TLinkAddress one = 1;
53             // Create
54             var createdLink = _linksGqlAdapter.CreateAndUpdate(one, i);
55             // Count
56             Assert.Equal(i, createdLink);
57             Assert.Equal(i, _linksGqlAdapter.Count());
58             var allLinks = new List<Link<TLinkAddress>>();
59             _linksGqlAdapter.Each(link =>
60             {
61                 allLinks.Add(new Link<TLinkAddress>(link));
62                 return _constants.Continue;
63             });
64             Assert.Equal(i, _linksGqlAdapter.Count());
65         }
66     }
67
68     [Fact]
69     public void CudTest()
70     {
71         TestCud();
72     }
73
74     [Fact]
75     public void EachTest()
76     {
77         TestCud();
78         var count = _linksGqlAdapter.Count();
79         TLinkAddress eachIterations = 0;
```

```

80     _linksGqlAdapter.Each(link =>
81     {
82         Assert.Equal(++eachIterations, _linksGqlAdapter.GetTarget(link));
83         return _linksGqlAdapter.Constants.Continue;
84     }, new Link<TLinkAddress>(_constants.Any, _constants.Any, _constants.Any));
85     Assert.Equal(count, eachIterations);
86 }
87 }

```

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

```

1  using GraphQL.Client.Http;
2  using GraphQL.Client.Serializer.Newtonsoft;
3  using Platform.Data.Doublets.Decorators;
4  using Platform.Data.Doublets.Gql.Client;
5  using Platform.Data.Doublets.Tests;
6  using Platform.IO;
7  using Platform.Memory;
8  using System;
9  using System.Collections.Generic;
10 using System.Diagnostics;
11 using System.IO;
12 using System.Net.Http;
13 using Xunit;
14 using TLinkAddress = System.UInt64;
15
16 namespace Platform.Data.Doublets.Gql.Tests;
17
18 public class DeepGenericLinksTests
19 {
20     public readonly string TempFilePath;
21     public readonly Uri EndPoint = new Uri("");
22
23     public DeepGenericLinksTests()
24     {
25         TempFilePath = new TemporaryFile();
26     }
27
28     [Fact (Skip = "Temp skip", Timeout = 60000)]
29     public void CRUDTest()
30     {
31         Using(links =>
32         {
33             var allLinks = links.All();
34             foreach (var linkToDelete in allLinks)
35             {
36                 var id = linkToDelete![0];
37                 if (links.Exists(id))
38                 {
39                     links.Delete(id);
40                 }
41             }
42             links.TestCRUDOperations();
43         });
44     }
45
46     [Fact (Skip = "Temp skip", Timeout = 60000)]
47     public void RawNumbersCRUDTest()
48     {
49         Using(links =>
50         {
51             var allLinks = links.All();
52             foreach (var linkToDelete in allLinks)
53             {
54                 var id = linkToDelete![0];
55                 if (links.Exists(id))
56                 {
57                     links.Delete(id);
58                 }
59             }
60             links.TestRawNumbersCRUDOperations();
61         });
62     }
63
64     [Fact (Skip = "Temp skip", Timeout = 60000)]
65     public void MultipleRandomCreationsAndDeletionsTest()
66     {
67         Using(links =>
68         {
69             var allLinks = links.All();
70             foreach (var linkToDelete in allLinks)

```

```

71         {
72             var id = linkToDelete![0];
73             if (links.Exists(id))
74             {
75                 links.Delete(id);
76             }
77         }
78         links.TestMultipleRandomCreationsAndDeletions(10);
79     });
80 }
81
82 [Fact (Skip = "Temp skip", Timeout = 60000)]
83 public void MultipleRandomCreationsAndDeletionsWithDecoratorsTest()
84 {
85     Using(links =>
86     {
87         var allLinks = links.All();
88         foreach (var linkToDelete in allLinks)
89         {
90             var id = linkToDelete![0];
91             if (links.Exists(id))
92             {
93                 links.Delete(id);
94             }
95         }
96         links.DecorateWithAutomaticUniquenessAndUsagesResolution().TestMultipleRandomCreationsAndDeletions(10);
97     });
98 }
99
100 private void Using(Action<ILinks<TLinkAddress>> action)
101 {
102     var graphqlClient = new GraphQLHttpClient(EndPoint, new NewtonsoftJsonSerializer());
103     var linksConstants = new LinksConstants<TLinkAddress>(true);
104     var token = "";
105     var linksGqlStorage = new DeepGqlAdapter(graphqlClient, linksConstants, token);
106     using var logFile = File.Open("linksLogger.txt", FileMode.Create, FileAccess.Write);
107     LoggingDecorator<TLinkAddress> decoratedLinksStorage = new(linksGqlStorage, logFile);
108     action(decoratedLinksStorage);
109 }
110 }

```

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

```

1  using GraphQL.Client.Http;
2  using GraphQL.Client.Serializer.Newtonsoft;
3  using Platform.Data.Doublets.Decorators;
4  using Platform.Data.Doublets.Gql.Client;
5  using Platform.Data.Doublets.Tests;
6  using Platform.IO;
7  using Platform.Memory;
8  using System;
9  using System.Diagnostics;
10 using System.IO;
11 using Xunit;
12
13 namespace Platform.Data.Doublets.Gql.Tests;
14
15 public class GenericLinksTests : IDisposable
16 {
17     public readonly string TempFilePath;
18     public readonly Uri EndPoint;
19     public readonly Process ServerProcess;
20
21     public GenericLinksTests()
22     {
23         TempFilePath = new TemporaryFile();
24         ServerProcess = TestExtensions.RunServer(TempFilePath);
25         EndPoint = TestExtensions.GetEndPointFromServerProcess(ServerProcess);
26     }
27
28     public void Dispose()
29     {
30         ServerProcess.Kill(true);
31     }
32
33     [Fact (Skip = "Temp skip", Timeout = 60000)]
34     public void CRUDTest()
35     {
36         Using(links => links.TestCRUDOperations());
37     }

```

```

38     [Fact (Skip = "Temp skip", Timeout = 60000)]
39     public void RawNumbersCRUDTest()
40     {
41         Using(links => links.TestRawNumbersCRUDOperations());
42     }
43
44     [Fact (Skip = "Temp skip", Timeout = 60000)]
45     public void MultipleRandomCreationsAndDeletionsTest()
46     {
47         Using(links => links.TestMultipleRandomCreationsAndDeletions(7));
48     }
49
50     [Fact (Skip = "Temp skip", Timeout = 60000)]
51     public void MultipleRandomCreationsAndDeletionsWithDecoratorsTest()
52     {
53         Using(links =>
54         {
55             var allLinks = links.All();
56             foreach (var linkToDelete in allLinks)
57             {
58                 var id = linkToDelete![0];
59                 if (links.Exists(id))
60                 {
61                     links.Delete(id);
62                 }
63             }
64             links.DecorateWithAutomaticUniquenessAndUsagesResolution().TestMultipleRandomCreationsAndDeletions(10);
65         });
66     }
67
68     private void Using(Action<ILinks<ulong>> action)
69     {
70         var graphqlClient = new GraphQLHttpClient(EndPoint, new Newtonsoft.Json.JsonSerializer());
71         var linksConstants = new LinksConstants<ulong>(true);
72         var linksGqlStorage = new LinksGqlAdapter(graphqlClient, linksConstants);
73         using var logFile = File.Open("linksLogger.txt", FileMode.Create, FileAccess.Write);
74         LoggingDecorator<ulong> decoratedLinksStorage = new(linksGqlStorage, logFile);
75         action(decoratedLinksStorage);
76     }
77 }

```

1.4 ./csharp/Platform.Data.Doublets.Gql.Tests/MutationTests.cs

```

1  using GraphQL;
2  using GraphQL.SystemTextJson;
3  using Newtonsoft.Json;
4  using Newtonsoft.Json.Linq;
5  using Platform.Data.Doublets.Gql.Client;
6  using Platform.Data.Doublets.Gql.Schema;
7  using Platform.Data.Doublets.Memory;
8  using Platform.Data.Doublets.Memory.United.Generic;
9  using Platform.IO;
10 using Platform.Memory;
11 using System;
12 using System.Collections.Generic;
13 using System.Linq;
14 using Xunit;
15 using TLinkAddress = System.UInt64;
16
17 namespace Platform.Data.Doublets.Gql.Tests
18 {
19     public class MutationTests
20     {
21         public static EqualityComparer<TLinkAddress> EqualityComparer =
22             EqualityComparer<TLinkAddress>.Default;
23         public static ILinks<ulong> CreateLinks() => CreateLinks<ulong>(new TemporaryFile());
24         public static ILinks<TLinkAddress> CreateLinks<TLinkAddress>(string dataDBFilename)
25         {
26             var linksConstants = new LinksConstants<TLinkAddress>(true);
27             return new UnitedMemoryLinks<TLinkAddress>(new
28                 FileMappedResizableDirectMemory(dataDBFilename),
29                 UnitedMemoryLinks<TLinkAddress>.DefaultLinksSizeStep, linksConstants,
30                 IndexTreeType.Default);
31         }
32
33         [Fact]
34         public void InsertLinksOne()
35         {

```

```

33     var links = CreateLinks();
34     LinksSchema linksSchema = new(links, new DefaultServiceProvider());
35     var jsonTask = linksSchema.ExecuteAsync(_ => { _.Query = @"
36 mutation {
37     insert_links_one(object: {from_id: 1, to_id: 1}) {
38         id
39         from_id
40         to_id
41     }
42 }
43 "; });
44     dynamic result =
45     ↪ Newtonsoft.Json.JsonConvert.DeserializeObject<dynamic>(jsonTask.Result);
46     if (result.ContainsKey("errors"))
47     {
48         throw new Exception(result.errors.ToString());
49     }
50
51 [Fact]
52 public void InsertLinks()
53 {
54     var links = CreateLinks();
55     LinksSchema linksSchema = new(links, new DefaultServiceProvider());
56     var jsonTask = linksSchema.ExecuteAsync(_ => { _.Query = @"
57 mutation {
58     insert_links(objects: [{ from_id: 1, to_id: 1 }, { from_id: 2, to_id: 2 }]) {
59         returning {
60             id
61             from_id
62             to_id
63         }
64     }
65 }
66 "; });
67     dynamic result =
68     ↪ Newtonsoft.Json.JsonConvert.DeserializeObject<dynamic>(jsonTask.Result);
69     if (result.ContainsKey("errors"))
70     {
71         throw new Exception(result.errors.ToString());
72     }
73
74 [Fact]
75 public void UpdateLinks()
76 {
77     var links = CreateLinks();
78     LinksSchema linksSchema = new(links, new DefaultServiceProvider());
79     var jsonTask = linksSchema.ExecuteAsync(_ => { _.Query = @"
80 mutation {
81     ↪ update_links(_set: { from_id: 1, to_id: 2 }, where: { from_id: { _eq: 2 }, to_id: {
82         ↪ _eq: 2 } }) {
83         returning {
84             id
85             from_id
86             to_id
87         }
88     }
89 }
90 "; });
91     dynamic result =
92     ↪ Newtonsoft.Json.JsonConvert.DeserializeObject<dynamic>(jsonTask.Result);
93     if (result.ContainsKey("errors"))
94     {
95         throw new Exception(result.errors.ToString());
96     }
97
98 [Fact]
99 public void DeleteLinks()
100 {
101     var links = CreateLinks();
102     LinksSchema linksSchema = new(links, new DefaultServiceProvider());
103     var jsonTask = linksSchema.ExecuteAsync(_ => { _.Query = @"
104 mutation {
105     delete_links(where: { from_id: { _eq: 1 }, to_id: { _eq: 1 } }) {
106         returning {
107             id

```

```

108         from_id
109         to_id
110     }
111 }
112 }
113 "; });
114 dynamic result =
115     ↪ Newtonsoft.Json.JsonConvert.DeserializeObject<dynamic>(jsonTask.Result);
116 if (result.ContainsKey("errors"))
117 {
118     throw new Exception(result.errors.ToString());
119 }
120
121 [Fact]
122 public void CreateZeroZeroAndUpdateToOneOneById()
123 {
124     var links = CreateLinks();
125     LinksSchema linksSchema = new(links, new DefaultServiceProvider());
126     var jsonTask = linksSchema.ExecuteAsync(_ => { _.Query = @"
127 mutation {
128     insert_links_one(object: {from_id: 0, to_id: 0}) {
129         id
130         from_id
131         to_id
132     }
133 }"; });
134 var jsonSerializer = new JsonSerializer();
135 var jsonResponse = jsonTask.Result;
136 Assert.False(JObject.Parse(jsonResponse).ContainsKey("errors"));
137 jsonTask = linksSchema.ExecuteAsync(_ => { _.Query = @"
138 mutation {
139     update_links(_set: { from_id: 1, to_id: 1 }, where: { id: {_eq: 1} }) {
140         returning {
141             id
142             from_id
143             to_id
144         }
145     }
146 }"; });
147 dynamic result =
148     ↪ Newtonsoft.Json.JsonConvert.DeserializeObject<dynamic>(jsonTask.Result);
149 if (result.ContainsKey("errors"))
150 {
151     throw new Exception(result.errors.ToString());
152 }
153 Assert.True(1 == Convert.ToInt32(result.data.update_links.returning[0].id));
154 }
155 }
156 }
157 }

```

1.5 ./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 }

```

```

24 [InlineData(@"
25 {
26     links {
27         id
28     }
29 }
30 ")
31 [InlineData(@"
32 {
33     links(
34         where: { id: { _eq: 1 }, from_id: { _eq: 1 }, to_id: { _eq: 1 } }
35         distinct_on: [from_id]
36         order_by: { id: asc }
37         offset: 0
38         limit: 1
39     ) {
40         id
41         from_id
42         from {
43             id
44             from_id
45             to_id
46         }
47         out {
48             id
49             from_id
50             to_id
51         }
52         to_id
53         to {
54             id
55             from_id
56             to_id
57         }
58         in {
59             id
60             from_id
61             to_id
62         }
63     }
64 }
65 ")
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 }

```

1.6 ./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 = $"run -f net5 {tempFilePath}",
19             ↪ RedirectStandardOutput = true, RedirectStandardInput = true};
20         var process = Process.Start(processStartInfo);
21         if (null == process || process.HasExited)
22         {
23             throw new Exception("Failed to start server process");
24         }
25         return process;
26     }
27
28     public static Uri GetEndPointFromServerProcess(Process process)
29     {
30         while (true)
31         {
32             var standartOutput = process?.StandardOutput;
33             if(standartOutput == null)
34             {
35                 Thread.Sleep(TimeSpan.FromSeconds(1));
36                 continue;
37             }
38             var processOutputLine = standartOutput.ReadLine();
39             if (string.IsNullOrEmpty(processOutputLine))
40             {
41                 Thread.Sleep(TimeSpan.FromSeconds(1));
42                 continue;
43             }
44             if (processOutputLine.Contains("Unable to start"))
45             {
46                 throw new Exception("Unable to start.");
47             }
48             if (processOutputLine.Contains("Now listening on: "))
49             {
50                 var index = processOutputLine.IndexOf("Now listening on: ",
51                     ↪ StringComparison.Ordinal) + "Now listening on:".Length;
52                 var uriString = processOutputLine.Substring(index);
53                 return new Uri($"{{uriString}}/v1/graphql");
54             }
55         }
56     }
57 }

```


Index

- ./csharp/Platform.Data.Doublets.Gql.Tests/ClientTests.cs, 1
- ./csharp/Platform.Data.Doublets.Gql.Tests/DeepGenericLinksTests.cs, 2
- ./csharp/Platform.Data.Doublets.Gql.Tests/GenericLinksTests.cs, 3
- ./csharp/Platform.Data.Doublets.Gql.Tests/MutationTests.cs, 4
- ./csharp/Platform.Data.Doublets.Gql.Tests/QueryTest.cs, 6
- ./csharp/Platform.Data.Doublets.Gql.Tests/TestExtensions.cs, 8