

LinksPlatform's Platform.RegularExpressions.Transformer.HasuraSQLSimplifier Class Library

1.1 ./csharp/Platform.RegularExpressions.Transformer.HasuraSQLSimplifier/HasuraSQLSimplifierTransformer.cs

```

1 using System.Collections.Generic;
2 using System.Linq;
3 using System.Text.RegularExpressions;
4
5 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
6
7 namespace Platform.RegularExpressions.Transformer.HasuraSQLSimplifier
8 {
9     /// <summary>
10     /// <para>
11     /// Represents the hasura sql simplifier transformer.
12     /// </para>
13     /// <para></para>
14     /// </summary>
15     /// <seealso cref="TextTransformer"/>
16     public class HasuraSQLSimplifierTransformer : TextTransformer
17     {
18         /// <summary>
19         /// <para>
20         /// The to list.
21         /// </para>
22         /// <para></para>
23         /// </summary>
24         public static readonly IList<ISubstitutionRule> DefaultRules = new List<SubstitutionRule>
25         {
26             // HTML clean up
27             (new Regex(@"<span class=""[^"""]*">(<[^>]*)</span>", "$1", 0),
28             // ('describe')
29             // 'describe'
30             (new Regex(@"\[\\s\\n]*('['']+')\[\\s\\n]*\\)", "$1", int.MaxValue),
31             // AND ('true' AND 'true')
32             //
33             (new Regex(@"\[\\s\\n]*AND\[\\s\\n]*\[\\s\\n]*'true' \[\\s\\n]*AND\[\\s\\n]*'true' \[\\s\\n]*\\)", "",
34             → 0),
35             // AND ('true')
36             //
37             (new Regex(@"\[\\s\\n]*AND\[\\s\\n]*'true'"), "", 0),
38             // ::
39             (new Regex(@"\[\\s]*::\[\\s]*"), "::", 0),
40             // ('describe'::text)
41             // 'describe'::text
42             (new Regex(@"\[\\s\\n]*('['']+'::text)\[\\s\\n]*\\)", "$1", 0),
43             // ("_0__be_0_nodes"."target_id")
44             // "_0__be_0_nodes"."target_id"
45             (new Regex(@"\[\\s\\n]*(""[^"""]*"")\[\\s\\n]*\.[\\s\\n]*(""[^"""]*"")\[\\s\\n]*\\)", "$1.$2",
46             → 0),
47             // ("public"."nodes"."_id")
48             // "public"."nodes"."_id"
49             (new Regex(@"\[\\s\\n]*(""[^"""]*"")\[\\s\\n]*\.[\\s\\n]*(""[^"""]*"")\[\\s\\n]*\.[\\s\\n]*(""[^"""]*"")\[\\s\\n]*\\)", "$1.$2.$3",
50             → 0),
51             // LIMIT\n\t\t\t\t\t1
52             // LIMIT 1
53             (new Regex(@"(LIMIT)\[\\s\\n]*([\\d]+)", "$1 $2", 0),
54             // ("_0__be_0_nodes"."type" = 'describe'::text)
55             // "_0__be_0_nodes"."type" = 'describe'::text
56             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
57             → int.MaxValue),
58             // (EXISTS (...))
59             // EXISTS (...)
60             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
61             → int.MaxValue),
62             // ((EXISTS (...)))
63             // EXISTS (...)
64             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
65             → int.MaxValue),
66             // ((EXISTS (...)))
67             // EXISTS (...)
68             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
69             → int.MaxValue),
70             // ((EXISTS (...)))
71             // EXISTS (...)
72             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
73             → int.MaxValue),
74             // ((EXISTS (...)))
75             // EXISTS (...)
76             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
77             → int.MaxValue),
78             // ((EXISTS (...)))
79             // EXISTS (...)
80             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
81             → int.MaxValue),
82             // ((EXISTS (...)))
83             // EXISTS (...)
84             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
85             → int.MaxValue),
86             // ((EXISTS (...)))
87             // EXISTS (...)
88             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
89             → int.MaxValue),
90             // ((EXISTS (...)))
91             // EXISTS (...)
92             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
93             → int.MaxValue),
94             // ((EXISTS (...)))
95             // EXISTS (...)
96             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
97             → int.MaxValue),
98             // ((EXISTS (...)))
99             // EXISTS (...)
100             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
101             → int.MaxValue),
102             // ((EXISTS (...)))
103             // EXISTS (...)
104             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
105             → int.MaxValue),
106             // ((EXISTS (...)))
107             // EXISTS (...)
108             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
109             → int.MaxValue),
110             // ((EXISTS (...)))
111             // EXISTS (...)
112             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
113             → int.MaxValue),
114             // ((EXISTS (...)))
115             // EXISTS (...)
116             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
117             → int.MaxValue),
118             // ((EXISTS (...)))
119             // EXISTS (...)
120             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
121             → int.MaxValue),
122             // ((EXISTS (...)))
123             // EXISTS (...)
124             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
125             → int.MaxValue),
126             // ((EXISTS (...)))
127             // EXISTS (...)
128             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
129             → int.MaxValue),
130             // ((EXISTS (...)))
131             // EXISTS (...)
132             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
133             → int.MaxValue),
134             // ((EXISTS (...)))
135             // EXISTS (...)
136             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
137             → int.MaxValue),
138             // ((EXISTS (...)))
139             // EXISTS (...)
140             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
141             → int.MaxValue),
142             // ((EXISTS (...)))
143             // EXISTS (...)
144             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
145             → int.MaxValue),
146             // ((EXISTS (...)))
147             // EXISTS (...)
148             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
149             → int.MaxValue),
150             // ((EXISTS (...)))
151             // EXISTS (...)
152             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
153             → int.MaxValue),
154             // ((EXISTS (...)))
155             // EXISTS (...)
156             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
157             → int.MaxValue),
158             // ((EXISTS (...)))
159             // EXISTS (...)
160             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
161             → int.MaxValue),
162             // ((EXISTS (...)))
163             // EXISTS (...)
164             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
165             → int.MaxValue),
166             // ((EXISTS (...)))
167             // EXISTS (...)
168             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
169             → int.MaxValue),
170             // ((EXISTS (...)))
171             // EXISTS (...)
172             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
173             → int.MaxValue),
174             // ((EXISTS (...)))
175             // EXISTS (...)
176             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
177             → int.MaxValue),
178             // ((EXISTS (...)))
179             // EXISTS (...)
180             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
181             → int.MaxValue),
182             // ((EXISTS (...)))
183             // EXISTS (...)
184             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
185             → int.MaxValue),
186             // ((EXISTS (...)))
187             // EXISTS (...)
188             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
189             → int.MaxValue),
190             // ((EXISTS (...)))
191             // EXISTS (...)
192             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
193             → int.MaxValue),
194             // ((EXISTS (...)))
195             // EXISTS (...)
196             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
197             → int.MaxValue),
198             // ((EXISTS (...)))
199             // EXISTS (...)
200             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
201             → int.MaxValue),
202             // ((EXISTS (...)))
203             // EXISTS (...)
204             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
205             → int.MaxValue),
206             // ((EXISTS (...)))
207             // EXISTS (...)
208             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
209             → int.MaxValue),
210             // ((EXISTS (...)))
211             // EXISTS (...)
212             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
213             → int.MaxValue),
214             // ((EXISTS (...)))
215             // EXISTS (...)
216             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
217             → int.MaxValue),
218             // ((EXISTS (...)))
219             // EXISTS (...)
220             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
221             → int.MaxValue),
222             // ((EXISTS (...)))
223             // EXISTS (...)
224             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
225             → int.MaxValue),
226             // ((EXISTS (...)))
227             // EXISTS (...)
228             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$2",
229             → int.MaxValue),
230             // ((EXISTS (...)))
231             // EXISTS (...)
232             (new Regex(@"(\\W)\\\[\\s\\n]*((?!SELECT) \[\\s\\n() \[\\s\\n]*\\)", "$1$
```

```

69     public HasuraSQLSimplifierTransformer()
70         : base(DefaultRules)
71     {
72     }
73 }
74 }

```

1.2 ./csharp/Platform.RegularExpressions.Transformer.HasuraSQLSimplifier.Tests/HasuraSQLSimplifierTransformerTests.cs

```

1  using Xunit;
2
3  namespace Platform.RegularExpressions.Transformer.HasuraSQLSimplifier.Tests
4  {
5      /// <summary>
6      /// <para>
7      /// Represents the hasura sql simplifier transformer tests.
8      /// </para>
9      /// <para></para>
10     /// </summary>
11     public class HasuraSQLSimplifierTransformerTests
12     {
13         /// <summary>
14         /// <para>
15         /// Tests that empty line test.
16         /// </para>
17         /// <para></para>
18         /// </summary>
19         [Fact]
20         public void EmptyLineTest()
21         {
22             // This test can help to test basic problems with regular expressions like incorrect
23             // ↪ syntax
24             var transformer = new HasuraSQLSimplifierTransformer();
25             var actualResult = transformer.Transform("");
26             Assert.Equal("", actualResult);
27         }
28
29         /// <summary>
30         /// <para>
31         /// Tests that basic request test.
32         /// </para>
33         /// <para></para>
34         /// </summary>
35         [Fact]
36         public void BasicRequestTest()
37         {
38             var original = @"SELECT
39 coalesce(json_agg("root"), '[]') AS "root"
40 FROM
41 (
42     SELECT
43         row_to_json(
44             (
45                 SELECT
46                     ""_2_e""
47                 FROM
48                     (
49                         SELECT
50                             ""_1_root.base"".id AS id""
51                         ) AS ""_2_e""
52             ) AS ""root""
53     FROM
54         (
55             SELECT
56                 *
57             FROM
58                 ""public"".nodes""
59             WHERE
60                 (
61                     (""public"".nodes.type") = (('auth_token') :: text)
62                 )
63             AND (
64                 EXISTS (
65                     SELECT
66                         1
67                     FROM
68                         ""public"".nodes AS ""_0__be_0_nodes""
69                     WHERE
70                         (
71

```

```

72         (
73             (
74                 ("_0__be_0_nodes"."_source_id") = ("public"."nodes"."_id")
75             )
76             AND ('true')
77         )
78         AND (
79             (
80                 (
81                     ("_0__be_0_nodes"."type") = (('describe') :: text))
82                     AND ('true')
83                 )
84                 AND (
85                     (
86                         (
87                             ("_0__be_0_nodes"."target_id") = (('X-Hasura-User-Id') :: text)
88                         )
89                         AND ('true')
90                     )
91                     AND ('true')
92                 )
93             )
94             AND (
95                 ('true')
96                 AND ('true')
97             )
98         )
99     )
100 )
101 )
102 ) AS ""_1_root.base""
103 LIMIT
104 1
105 ) AS ""_3_root"";
106
107     var expected = @"SELECT
108 coalesce(json_agg(""root""), '[]') AS ""root""
109 FROM
110 (
111     SELECT
112         row_to_json(
113             (
114                 SELECT
115                     ""_2_e""
116                 FROM
117                     (
118                         SELECT
119                             ""_1_root.base"."id"" AS ""id""
120                         ) AS ""_2_e""
121                     )
122             ) AS ""root""
123         FROM
124         (
125             SELECT
126                 *
127             FROM
128                 ""public"."nodes""
129             WHERE
130                 ""public"."nodes"."type"" = 'auth_token'::text
131                 AND EXISTS (
132                     SELECT
133                         1
134                     FROM
135                         ""public"."nodes"" AS ""_0__be_0_nodes""
136                     WHERE
137                         ""_0__be_0_nodes"."_source_id"" = ""public"."nodes"."_id""
138                         AND ""_0__be_0_nodes"."type"" = 'describe'::text
139                         AND ""_0__be_0_nodes"."target_id"" = 'X-Hasura-User-Id'::text
140                     )
141                 ) AS ""_1_root.base""
142             LIMIT 1
143         ) AS ""_3_root"";
144     var transformer = new HasuraSQLSimplifierTransformer();
145     var actual = transformer.Transform(original);
146     Assert.Equal(expected, actual);
147 }
148 }
149 }
150 }

```

Index

- ./csharp/Platform.RegularExpressions.Transformer.HasuraSQLSimplifier.Tests/HasuraSQLSimplifierTransformerTests.cs, 2
- ./csharp/Platform.RegularExpressions.Transformer.HasuraSQLSimplifier/HasuraSQLSimplifierTransformer.cs, 1