```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Data.SqlTypes;
 4  using Microsoft.SqlServer.Server;
 5
 6  namespace CLRUDF
 7  {
 8      public class Speech
 9      {
10          public static readonly string[] SPLIT_PARTS = {
11              "FIRSTNAME=", "LASTNAME=", "MONTH=", "DAY=",
12              "YEAR=", "WEBLINK=", "FILELINK=", "SPEECH=" };
13
14          private Dictionary<string, int> DocumentFrequency = new
              Dictionary<string, int>();
15
16          public string Firstname { get; set; }
17
18          public string Lastname { get; set; }
19
20          public string Month { get; set; }
21
22          public int Day { get; set; }
23
24          public int Year { get; set; }
25
26          public string Weblink { get; set; }
27
28          public string Filelink { get; set; }
29
30          public string Text { get; set; }
31
32          public bool IsNull { get; private set; }
33
34          public static Speech Null
35          {
36              get
37              {
38                  Speech speech = new Speech
39                  {
40                      IsNull = true
41                  };
42                  return speech;
43              }
44          }
45
46          public override string ToString()
47          {
48              return $"FIRSTNAME={Firstname}LASTNAME={Lastname}MONTH={Month}" +
```

```
49                  $"DAY={Day}YEAR={Year}WEBLINK={Weblink}FILELINK={Filelink}" +
50                  $"SPEECH={Text}";
51          }
52
53          [SqlMethod(OnNullCall = false)]
54          public static Speech Parse(SqlString s)
55          {
56              if (s.IsNull)
57              {
58                  return Null;
59              }
60
61              string[] parts = s.ToString().Split(SPLIT_PARTS,
                  StringSplitOptions.None);
62
63              if (parts.Length < 8)
64              {
65                  return Null;
66              }
67
68              if (!int.TryParse(parts[3], out int day))
69              {
70                  return Null;
71              }
72
73              if (!int.TryParse(parts[4], out int year))
74              {
75                  return Null;
76              }
77
78              Speech speech = new Speech
79              {
80                  IsNull = false,
81                  Firstname = parts[0],
82                  Lastname = parts[1],
83                  Month = parts[2],
84                  Day = day,
85                  Year = year,
86                  Weblink = parts[5],
87                  Filelink = parts[6],
88                  Text = parts[7]
89              };
90              return speech;
91          }
92
93          public void UpdateFrequency(string word)
94          {
95              if (DocumentFrequency.ContainsKey(word))
96              {
```

```
 97                 DocumentFrequency[word]++;
 98             }
 99             else
100             {
101                 DocumentFrequency[word] = 1;
102             }
103         }
104
105         public override bool Equals(object obj)
106         {
107             if (!(obj is Speech))
108             {
109                 return false;
110             }
111
112             var speech = (Speech)obj;
113             return Firstname == speech.Firstname &&
114                    Lastname == speech.Lastname &&
115                    Month == speech.Month &&
116                    Day == speech.Day &&
117                    Year == speech.Year;
118         }
119
120         public override int GetHashCode()
121         {
122             var hashCode = 480983858;
123             hashCode = hashCode * -1521134295 +                                ↵
                EqualityComparer<string>.Default.GetHashCode(Firstname);
124             hashCode = hashCode * -1521134295 +                                ↵
                EqualityComparer<string>.Default.GetHashCode(Lastname);
125             hashCode = hashCode * -1521134295 +                                ↵
                EqualityComparer<string>.Default.GetHashCode(Month);
126             hashCode = hashCode * -1521134295 + Day.GetHashCode();
127             hashCode = hashCode * -1521134295 + Year.GetHashCode();
128             return hashCode;
129         }
130     }
131 }
132
```