```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Data;
 4  using System.Linq;
 5  using System.Data.Linq.Mapping;
 6  using System.Xml.Linq;
 7  using System.Collections;
 8  using Sfinx;
 9  using static Vertigo.DAL.Functions;
10
11  namespace Vertigo.DAL
12  {
13      [Table(Name = "u_money")]
14      public class Umoney
15      {
16          /// <summary>
17          /// Copyrigts by KBSOFT
18          /// </summary>
19          #region Properties
20          [Column(Storage = "_id", DbType = "int", IsPrimaryKey = true,
              CanBeNull = false)]
21          public Int32 Id { get { return this._id; } set { this._id = value; } }
22          private Int32 _id;
23
24          [Column(Storage = "_status", DbType = "char", CanBeNull = false)]
25          public String Status { get { return this._status; } set { this._status
                = value; } }
26          private String _status;
27
28          [Column(Storage = "_ext", DbType = "char", CanBeNull = false)]
29          public String Ext { get { return this._ext; } set { this._ext =
              value; } }
30          private String _ext = "Z";
31
32          [Column(Storage = "_u_contract_id", DbType = "int", CanBeNull =
              false)]
33          public Int32 U_Contract_Id { get { return this._u_contract_id; } set
              { this._u_contract_id = value; } }
34          private Int32 _u_contract_id;
35
36          [Column(Storage = "_value", DbType = "money", CanBeNull = false)]
37          public Decimal Value { get { return this._value; } set { this._value =
                value.SQLcheck(); } }
38          private Decimal _value;
39
40          [Column(Storage = "_kind", DbType = "varchar", CanBeNull = false)]
41          public String Kind { get { return this._kind; } set { this._kind =
              value; } }
42          private String _kind = KindEnum.PLN.ToString();
43
44          [Column(Storage = "_period", DbType = "varchar", CanBeNull = false)]
45          public String Period { get { return this._period; } set { this._period
                = value; } }
46          private String _period = PeriodEnum.rok.ToString();
47
48          [Column(Storage = "_description", DbType = "varchar", CanBeNull =
```

```
            false)]
49          public String Description { get { return this._description; } set
              { this._description = value.SQLcheck(); } }
50          private String _description="";
51          #endregion
52
53          public List<Umoney> Data;
54          public IEnumerator GetEnumerator()
55          {
56              for (int i = 0; i <= Data.Count() - 1; i++)
57              {
58                  yield return Data.ElementAt(i);
59              }
60          }
61
62          public Umoney()
63          { }
64
65          public Umoney(string SQLEnd):this(SQLEnd,null)
66          { }
67
68          public Umoney(string SQLEnd, dbcn cn)
69          {
70              CheckCN(ref cn);
71              try
72              {
73                  XDocument doc = XDocument.Parse(cn.GetXML(cn.ExecuteSQL
                      ("SELECT * FROM [vertigo].[dbo].[u_money] " + SQLEnd)));
74                  XNamespace z = "#RowsetSchema";
75                  Data = (from lst in doc.Descendants(z + "row")
76                          select new Umoney
77                          {
78                              Id = String.IsNullOrEmpty(lst.Attribute
                        ("id").Value) ? 0 : Convert.ToInt32(lst.Attribute
                        ("id").Value),
79                              Status = String.IsNullOrEmpty(lst.Attribute
                        ("status").Value) ? "" : lst.Attribute("status").Value,
80                              Ext = String.IsNullOrEmpty(lst.Attribute
                        ("ext").Value) ? "" : lst.Attribute("ext").Value,
81                              U_Contract_Id = String.IsNullOrEmpty(lst.Attribute
                        ("u_contract_id").Value) ? 0 : Convert.ToInt32
                        (lst.Attribute("u_contract_id").Value),
82                              Value = String.IsNullOrEmpty(lst.Attribute
                        ("value").Value) ? 0 : Convert.ToDecimal(lst.Attribute
                        ("value").Value.Replace(".",",")),
83                              Kind = String.IsNullOrEmpty(lst.Attribute
                        ("kind").Value) ? "" : lst.Attribute("kind").Value,
84                              Period = String.IsNullOrEmpty(lst.Attribute
                        ("period").Value) ? "" : lst.Attribute("period").Value,
85                              Description = String.IsNullOrEmpty(lst.Attribute
                        ("description").Value) ? "" : lst.Attribute
                        ("description").Value
86                          }).ToList();
87              }
88              catch (Exception ex)
89              { throw new Exception(ex.Message + ", " + cn.ErrMess); }
```

```
 90          }
 91
 92
 93          public enum ExtEnum { Z, P }
 94          public enum KindEnum { PLN, EU, USD}
 95          public enum PeriodEnum { tydz, mies, kwart, półr, rok}
 96
 97          public int Insert(dbcn cn=null)
 98          {
 99              CheckCN(ref cn);
100              int ret = 0;
101              try
102              {
103                  if (IsValidInsert())
104                  {
105                      string sql = $"INSERT INTO [dbo].[u_money] ([status],
                        [ext],[u_contract_id],[value],[kind],[period],
                        [description]) VALUES " +
106                       $"('{Status}','{Ext}',{U_Contract_Id},'{Value.ToString
                        ().Replace
                        (",",".")}','{Kind}','{Period}','{Description}')";
107                      ret = cn.ExecuteSQLscope(sql,true);
108                  }
109              }
110              catch (Exception ex)
111              {
112                  throw new Exception("Nie można dodać danych do bazy. " +
                      ex.Message);
113              }
114              return ret;
115          }
116
117          public void Update(dbcn cn=null)
118          {
119              CheckCN(ref cn);
120              try
121              {
122                  if (IsValidUpdate())
123                  {
124                      string sql = $"UPDATE [dbo].[u_money] SET [status] =
                        '{Status}',[u_contract_id] = {U_Contract_Id} ,[value] =
                        '{Value.ToString().Replace(",", ".")}'" +
125                          $" ,[kind] = '{Kind}' ,[period] = '{Period}' ,
                        [description] = '{Description}' WHERE id=" + Id;
126                      cn.ExecuteSQL(sql);
127                  }
128              }
129              catch (Exception ex)
130              {
131                  throw new Exception("Nie można dodać danych do bazy. " +
                      ex.Message);
132              }
133          }
134
135          public void Delete(dbcn cn=null)
136          {
```

```
137             CheckCN(ref cn);
138             try
139             {
140                 if (IsValidDelete())
141                 {
142                     string sql = $"DELETE FROM [dbo].[u_money] WHERE id= " +
                        Id;
143                     cn.ExecuteSQL(sql);
144                 }
145             }
146             catch (Exception ex)
147             {
148                 throw new Exception("Nie można usunąć danych z bazy. " +
                      ex.Message);
149             }
150         }
151
152
153         public bool IsValidInsert()
154         {
155             if (Status is null || Kind is null || Period is null ||
                  Description is null) { throw new Exception("Uzupełnij dane"); }
156             if (Status.Trim().Length != 1) { throw new Exception("Status
                  danych jest nieprawidłowy. " + Status); }
157             if (!System.Enum.GetNames(typeof(ExtEnum)).Contains(Ext)) { throw
                  new Exception("Rodzaj słownika jest nieprawidłowy. " + Ext); }
158             if (U_Contract_Id == 0) { throw new Exception("Identyfikator
                  pozycji kontraktu nie jest powiązany z danymi!"); }
159             return true;
160         }
161
162         public bool IsValidUpdate()
163         {
164             IsValidInsert();
165             if (Id <1) { throw new Exception("Identyfikator pozycji nie jest
                  powiązany z danymi!"); }
166             return true;
167         }
168
169         private bool IsValidDelete()
170         {
171             if (Id < 1) { throw new Exception("Brak identyfikatora pozycji do
                  usunięcia!"); }
172             return true;
173         }
174
175
176     }
177
178 }
179
180
```