

Solution Datamodel (5 Projects).....	4
Table of Contents	1
WATB.Datamodel.Context	4
GenericDatamodel.cs	4
WATB.Datamodel.Context	4
GenericDatamodel	4
AddAsset [Method]	5
AddBaseContract [Method]	5
AddChild [Method]	4
AddContent [Method]	4
AddContract [Method]	5
AddItem [Method]	4
AddLink [Method]	5
AddLink [Method]	5
AddLink [Method]	5
AddProperty [Method]	4
AddReceivedLink [Method]	5
AddReceivedLink [Method]	5
AddReceivedLink [Method]	5
AddRelation [Method]	4
AddSentLink [Method]	5
AddSentLink [Method]	5
AddSentLink [Method]	5
AddTemplate [Method]	4
AmountLinks [Property]	4
Assets [Property]	4
Children [Property]	4
Contents [Property]	4
Contracts [Property]	4
CountLinks [Property]	4
GenericDatamodel [Method]	4
GenericDatamodel [Method]	4
Identifiers [Property]	4
Items [Property]	4
Links [Property]	4
OwnedContracts [Property]	4
Properties [Property]	4
Relations [Property]	4
Templates [Property]	4
WATB.Datamodel.Factory	7
GenericFactory.cs	7
WATB.Datamodel.Factory	7
GenericFactory	7
Initialize [Method]	7
Initialize [Method]	7
Initialize [Method]	7
Initialize [Method]	7
Initialize [Method]	7
Initialize [Method]	7
Initialize [Method]	8
Initialize [Method]	8
Initialize [Method]	8
Initialize [Method]	8
Initialize [Method]	8
Initialize [Method]	8
Initialize [Method]	8
Initialize [Method]	9
Initialize [Method]	9
WATB.Datamodel.Generic	10
AmountLink.cs	10
WATB.Datamodel.Generic	10
AmountLink	10
Amount [Property]	10
ToString [Method]	10
Asset.cs	11
WATB.Datamodel.Generic	11
Asset	11
AddOwnerLink [Method]	11
Links [Property]	11
ToString [Method]	11
BaseContract.cs	12
WATB.Datamodel.Generic	12
BaseContract	12
AddTransaction [Method]	12
Status [Property]	12
ToString [Method]	12
Transactions [Property]	12
Baseltem.cs	13
WATB.Datamodel.Generic	13
Baseltem	13
AddChild [Method]	13
AddOwnedContract [Method]	13
AddProductLink [Method]	13

AddProperty [Method]	13
AddRelation [Method]	13
Children [Property]	13
DeleteRelation [Method]	14
Name [Property]	13
OwnedContracts [Property]	13
OwnedLinks [Property]	13
Properties [Property]	13
Relations [Property]	13
ToString [Method]	13
BaseLink.cs	15
WATB.Datamodel.Generic	15
BaseLink	15
Asset [Property]	15
Contract [Property]	15
LinkType [Property]	15
Master [Property]	15
TimeStamp [Property]	15
ToString [Method]	15
ChildItem.cs	16
WATB.Datamodel.Generic	16
ChildItem	16
Parent [Property]	16
ToString [Method]	16
Contract.cs	17
WATB.Datamodel.Generic	17
Contract	17
Owner [Property]	17
ToString [Method]	17
ContractStatus.cs	18
WATB.Datamodel.Generic	18
CountLink.cs	19
WATB.Datamodel.Generic	19
CountLink	19
Count [Property]	19
ToString [Method]	19
DataContent.cs	20
WATB.Datamodel.Generic	20
DataContent	20
_content_Changed [Method]	20
Content [Property]	20
ContentString [Property]	20
Template [Property]	20
ToString [Method]	20
Validate [Method]	20
DataTemplate.cs	21
WATB.Datamodel.Generic	21
DataTemplate	21
_schema_Changed [Method]	22
_stylesheet_Changed [Method]	22
Contents [Property]	21
Schema [Property]	21
SchemaString [Property]	21
Stylesheet [Property]	21
StylesheetString [Property]	21
ToString [Method]	21
Validate [Method]	22
ValidateSchema [Method]	22
ValidateStylesheet [Method]	22
Identifier.cs	23
WATB.Datamodel.Generic	23
Identifier	23
Id [Property]	23
ToString [Method]	23
LinkType.cs	24
WATB.Datamodel.Generic	24
Property.cs	25
WATB.Datamodel.Generic	25
Property	25
Owner [Property]	25
ToString [Method]	25
Relation.cs	26
WATB.Datamodel.Generic	26
Relation	26
Add [Method]	26
Delete [Method]	26
Items [Property]	26
ToString [Method]	26
WATB.Datamodel.Testing	27
UnitTestContract.cs	27
WATB.Datamodel.Testing	27

UnitTestContract.....	27
ClassInit [Method]	27
Db [Property].....	27
Dispose [Method].....	27
Initialize [Method].....	27
TestMakeContract [Method]	28
TestMakeOwnedContract [Method].....	28
WhatIsInTheDatabase [Method].....	27
UnitTestItemFactory.cs.....	30
WATB.Datamodel.Testing.....	30
UnitTestItemFactory	30
ClassInit [Method]	30
Db [Property].....	30
Dispose [Method].....	30
Initialize [Method].....	30
TestDeleteProperty [Method].....	33
TestDeleteRelation [Method].....	32
TestMakeChild [Method].....	31
TestMakeContent [Method]	31
TestMakeEmptyContract [Method].....	33
TestMakeEmptyOwnedContract [Method].....	33
TestMakeEmptyProperty [Method]	32
TestMakeEmptyRelation [Method].....	31
TestMakeFilledProperty [Method].....	32
TestMakeFilledRelation [Method]	32
TestMakeItem [Method].....	31
TestMakeSubject [Method].....	33
TestMakeTemplate [Method].....	31
WhatIsInTheDatabase [Method].....	30
UnitTestLinkItems.cs	35
WATB.Datamodel.Testing.....	35
UnitTestLinkItems	35
ClassInit [Method]	35
Db [Property].....	35
Dispose [Method].....	35
Initialize [Method].....	35
TestMakeFloatLink [Method]	36
TestMakeIntegerLink [Method]	36
TestMakeLink [Method]	36
WhatIsInTheDatabase [Method].....	35

```
14
15  using System.Data.Entity;
16  using System.Xml.Linq;
17  using WATB.Datamodel.Factory;
18  using WATB.Datamodel.Generic;
19
20  namespace WATB.Datamodel.Context
21  {
25  public class GenericDatamodel : DbContext
26  {
30      public GenericDatamodel() : this("name=Datamodel") { }
31
36      public GenericDatamodel(string datamodel) : base(datamodel) { }
37
42      public virtual DbSet<Identifier> Identifiers { get; set; }
43
48      public virtual DbSet<DataContent> Contents { get; set; }
49
54      public virtual DbSet<DataTemplate> Templates { get; set; }
55
60      public virtual DbSet<BaseItem> Items { get; set; }
61
66      public virtual DbSet<ChildItem> Children { get; set; }
67
72      public virtual DbSet<Relation> Relations { get; set; }
73
78      public virtual DbSet<Property> Properties { get; set; }
79
84      public virtual DbSet<Asset> Assets { get; set; }
85
90      public virtual DbSet<BaseContract> Contracts { get; set; }
91
96      public virtual DbSet<Contract> OwnedContracts { get; set; }
97
98
99
104     public virtual DbSet<BaseLink> Links { get; set; }
105
110     public virtual DbSet<CountLink> CountLinks { get; set; }
111
116     public virtual DbSet<AmountLink> AmountLinks { get; set; }
117
124     public DataContent AddContent(XElement content, DataTemplate template) {
        return Contents.Add(Contents.Create().Initialize(content, template)); }
125
132     public DataTemplate AddTemplate(XElement schema, XElement stylesheet) {
        return Templates.Add(Templates.Create().Initialize(schema, stylesheet));
    }
133
141     public BaseItem AddItem(string name, XElement content = null,
        DataTemplate template = null) { return Items.Add(Items.Create().
        Initialize(name, content, template)); }
142
151     public ChildItem AddChild(string name, BaseItem parent, XElement content
        = null, DataTemplate template = null) { return Children.Add(Children.
        Create().Initialize(name, parent, content, template)); }
152
161     public Relation AddRelation(string name, XElement content = null,
        DataTemplate template = null, params BaseItem[] items) { return
        Relations.Add(Relations.Create().Initialize(name, content, template,
        items)); }
162
172     public Property AddProperty(string name, BaseItem owner, XElement
```

```
1      2
content = null, DataTemplate template = null, params BaseItem[] items) {
    return Properties.Add(Properties.Create().Initialize(name, owner,
content, template, items)); }
173
182    public Asset AddAsset(string name, XElement content = null, DataTemplate
template = null, params BaseLink[] contracts) { return Assets.Add(
Assets.Create().Initialize(name, content, template, contracts)); }
183
193    public BaseContract AddBaseContract(string name, ContractStatus status,
XElement content = null, DataTemplate template = null, params BaseLink[]
contracts) { return Contracts.Add(Contracts.Create().Initialize(name,
status, content, template, contracts)); }
194
205    public Contract AddContract(string name, BaseItem owner, ContractStatus
status, XElement content = null, DataTemplate template = null, params
BaseLink[] contracts) { return OwnedContracts.Add(OwnedContracts.Create(
).Initialize(name, owner, status, content, template, contracts)); }
206
217    public BaseLink AddLink(LinkType linkType, BaseContract baseContract,
BaseItem owner, Asset asset, XElement content = null, DataTemplate
template = null) { return Links.Add(Links.Create().Initialize(linkType,
baseContract, owner, asset, content, template)); }
218
230    public BaseLink AddLink(LinkType linkType, BaseContract baseContract,
BaseItem owner, Asset asset, int count, XElement content = null,
DataTemplate template = null) { return CountLinks.Add(CountLinks.Create(
).Initialize(linkType, baseContract, owner, asset, count, content,
template)); }
231
243    public BaseLink AddLink(LinkType linkType, BaseContract baseContract,
BaseItem owner, Asset asset, double amount, XElement content = null,
DataTemplate template = null) { return AmountLinks.Add(AmountLinks.
Create().Initialize(linkType, baseContract, owner, asset, amount,
content, template)); }
244
254    public BaseLink AddSentLink(BaseContract baseContract, BaseItem owner,
Asset asset, XElement content = null, DataTemplate template = null) {
return Links.Add(Links.Create().Initialize(LinkType.Send, baseContract,
owner, asset, content, template)); }
255
266    public BaseLink AddSentLink(BaseContract baseContract, BaseItem owner,
Asset asset, int count, XElement content = null, DataTemplate template =
null) { return CountLinks.Add(CountLinks.Create().Initialize(LinkType.
Send, baseContract, owner, asset, count, content, template)); }
267
278    public BaseLink AddSentLink(BaseContract baseContract, BaseItem owner,
Asset asset, double amount, XElement content = null, DataTemplate
template = null) { return AmountLinks.Add(AmountLinks.Create().
Initialize(LinkType.Send, baseContract, owner, asset, amount, content,
template)); }
279
289    public BaseLink AddReceivedLink(BaseContract baseContract, BaseItem
owner, Asset asset, XElement content = null, DataTemplate template =
null) { return Links.Add(Links.Create().Initialize(LinkType.Receive,
baseContract, owner, asset, content, template)); }
290
301    public BaseLink AddReceivedLink(BaseContract baseContract, BaseItem
owner, Asset asset, int count, XElement content = null, DataTemplate
template = null) { return CountLinks.Add(CountLinks.Create().Initialize(
LinkType.Receive, baseContract, owner, asset, count, content, template)
; }
302
313    public BaseLink AddReceivedLink(BaseContract baseContract, BaseItem
owner, Asset asset, double amount, XElement content = null, DataTemplate
```

```
1      template = null) { return AmountLinks.Add(AmountLinks.Create().  
2      Initialize(LinkType.Receive, baseContract, owner, asset, amount, content  
      , template)); }  
314  
315 }
```

```
14
15  using System;
16  using System.Collections.Generic;
17  using System.Linq;
18  using System.Xml.Linq;
19  using WATB.Datamodel.Generic;
20
21  namespace WATB.Datamodel.Factory
22  {
26      public static class GenericFactory
27      {
34          public static T Initialize<T>(this T item) where T : Identifier
35          {
36              T result = item;
37              result.Id = Guid.NewGuid();
38              return result;
39          }
40
49          public static T Initialize<T>(this T item, XElement content,
50          DataTemplate template) where T : DataContent
51          {
52              T result = Initialize(item);
53              result.Content = content;
54              result.Template = template;
55              return result;
56          }
65          public static T Initialize<T>(this T item, XElement schema, XElement
66          stylesheet) where T : DataTemplate
67          {
68              T result = Initialize(item);
69              result.Contents = new HashSet<DataContent>();
70              result.Schema = schema;
71              result.Stylesheet = stylesheet;
72              return result;
73          }
83          public static T Initialize<T>(this T item, string name, XElement content
84          = null, DataTemplate template = null) where T : BaseItem
85          {
86              T result = Initialize(item, content, template);
87              result.Name = name;
88              result.Children = new HashSet<ChildItem>();
89              result.OwnedLinks = new HashSet<BaseLink>();
90              result.Relations = new HashSet<Relation>();
91              result.OwnedContracts = new HashSet<Contract>();
92              result.Properties = new HashSet<Property>();
93              return result;
94          }
105         public static T Initialize<T>(this T item, string name, BaseItem parent,
106         XElement content = null, DataTemplate template = null) where T :
107         ChildItem
108         {
109             T result = Initialize(item, name, content, template);
110             parent.AddChild(result);
111             return result;
112         }
122         public static T Initialize<T>(this T item, string name, XElement content
123         = null, DataTemplate template = null, params BaseItem[] relations)
124         where T : Relation
125         {
126             T result = Initialize(item, name, content, template);
```

```
1      2      3
125     result.Items = new HashSet<BaseItem>();
126     foreach (BaseItem relation in relations.OrderBy(i => i.Name).ToList
      ()) { relation.AddRelation(result); }
127     return result;
128 }
129
141     public static T Initialize<T>(this T item, string name, BaseItem owner,
      XElement content = null, DataTemplate template = null, params BaseItem[]
      items) where T : Property
142     {
143         T result = Initialize(item, name, content, template, items);
144         owner.AddProperty(result);
145         return result;
146     }
147
158     public static T Initialize<T>(this T item, string name, XElement content
      = null, DataTemplate template = null, params BaseLink[] contracts)
      where T : Asset
159     {
160         T result = Initialize(item, name, content, template);
161         result.Links = new HashSet<BaseLink>();
162         foreach (BaseLink contract in contracts.OrderBy(i => i.TimeStamp).
      ToList()) { result.AddOwnerLink(contract); }
163         return result;
164     }
165
177     public static T Initialize<T>(this T item, string name, ContractStatus
      status, XElement content = null, DataTemplate template = null, params
      BaseLink[] contracts) where T : BaseContract
178     {
179         T result = Initialize(item, name, content, template, contracts);
180         result.Status = status;
181         result.Transactions = new HashSet<BaseLink>();
182         return result;
183     }
184
197     public static T Initialize<T>(this T item, string name, BaseItem owner,
      ContractStatus status, XElement content = null, DataTemplate template =
      null, params BaseLink[] contracts) where T : Contract
198     {
199         T result = Initialize(item, name, status, content, template,
      contracts);
200         owner.AddOwnedContract(item);
201         return result;
202     }
203
216     public static T Initialize<T>(this T link, LinkType linkType,
      BaseContract baseContract, BaseItem owner, Asset product, XElement
      content = null, DataTemplate template = null) where T : BaseLink
217     {
218         T result = Initialize(link, content, template);
219         result.Id = Guid.NewGuid();
220         result.TimeStamp = DateTime.Now;
221         result.LinkType = linkType;
222         baseContract.AddTransaction(result);
223         owner.AddProductLink(result);
224         product.AddOwnerLink(result);
225         return result;
226     }
227
241     public static T Initialize<T>(this T link, LinkType linkType,
      BaseContract baseContract, BaseItem owner, Asset product, double amount,
      XElement content = null, DataTemplate template = null) where T :
      AmountLink
1      2
```

```
242 | 1 2 |  
243 | | {  
244 | |     T result = Initialize(link, linkType, baseContract, owner, product,  
245 | |     content, template);  
246 | |     result.Amount = amount;  
247 | |     return result;  
248 | | }  
249 | |  
250 | | public static T Initialize<T>(this T link, LinkType linkType,  
251 | | BaseContract baseContract, BaseItem owner, Asset product, int count,  
252 | | XElement content = null, DataTemplate template = null) where T :  
253 | | CountLink  
254 | | {  
255 | |     T result = Initialize(link, linkType, baseContract, owner, product,  
256 | |     content, template);  
257 | |     result.Count = count;  
258 | |     return result;  
259 | | }  
260 | | }  
261 | | }
```

```
14
15  using System.ComponentModel.DataAnnotations;
16  using System.ComponentModel.DataAnnotations.Schema;
17
18  namespace WATB.Datamodel.Generic
19  {
24      [Table("AmountLinks")]
25      public class AmountLink : BaseLink
26      {
31          public override string ToString()
32          {
33              return string.Format("{0}\tAmount: {1:0,000.00###}", base.ToString(
34                  ), Amount);
39          [Required(ErrorMessage = "Need to specify amount of subject items.")]
40          public double Amount { get; set; }
41      }
42  }
```

```
14
15  using System.Collections.Generic;
16  using System.ComponentModel.DataAnnotations.Schema;
17
18  namespace WATB.Datamodel.Generic
19  {
23      [Table("Assets")]
24      public class Asset : BaseItem
25      {
30          public override string ToString()
31          {
32              return string.Format("{0}\tAsset links: {1}\r\n", base.ToString(),
33                                     Links.Count);
34          }
39          [InverseProperty("Asset")]
40          public virtual ICollection<BaseLink> Links { get; set; }
41
46          public void AddOwnerLink(BaseLink link)
47          {
48              Links.Add(link);
49              link.Asset = this;
50          }
51      }
52  }
```

```
14
15  using System.Collections.Generic;
16  using System.ComponentModel.DataAnnotations;
17  using System.ComponentModel.DataAnnotations.Schema;
18
19  namespace WATB.Datamodel.Generic
20  {
24      [Table("Contracts")]
25      public class BaseContract : Asset
26      {
31          public override string ToString()
32          {
33              return string.Format(
34                  "{0}\tContract status {1} with {2} transactions\r\n", base.ToString(
35                      ), Status, Transactions.Count);
36          }
37
40          [Required]
41          public ContractStatus Status { get; set; }
42
47          [InverseProperty("Contract")]
48          public virtual ICollection<BaseLink> Transactions { get; set; }
49
54          public void AddTransaction(BaseLink link)
55          {
56              Transactions.Add(link);
57              link.Contract = this;
58          }
59      }
60 }
```

```
14
15  using System.Collections.Generic;
16  using System.ComponentModel.DataAnnotations;
17  using System.ComponentModel.DataAnnotations.Schema;
18
19  namespace WATB.Datamodel.Generic
20  {
24      [Table("Items")]
25      public class BaseItem : DataContent
26      {
31          public override string ToString()
32          {
33              return string.Format(
34                  "{0}\tName: {1}\r\n\tOwned links: {2} links\r\n\tChildren: {3}\r\n\t
35                  Properties: {4}\r\n\tOwned contracts: {5}\r\n\tRelations: {6}\r\n",
36                  base.ToString(), Name, OwnedLinks.Count, Children.Count,
37                  Properties.Count, OwnedContracts.Count, Relations.Count);
38          }
39          [Required(ErrorMessage = "Name is required.")]
40          [MaxLength(240)]
41          public string Name { get; set; }
42
43          [InverseProperty("Master")]
44          public virtual ICollection<BaseLink> OwnedLinks { get; set; }
45
46          [InverseProperty("Parent")]
47          public virtual ICollection<ChildItem> Children { get; set; }
48
49          [InverseProperty("Owner")]
50          public virtual ICollection<Property> Properties { get; set; }
51
52          [InverseProperty("Owner")]
53          public virtual ICollection<Contract> OwnedContracts { get; set; }
54
55          [InverseProperty("Items")]
56          public virtual ICollection<Relation> Relations { get; set; }
57
58          public void AddProductLink(BaseLink link)
59          {
60              OwnedLinks.Add(link);
61              link.Master = this;
62          }
63
64          public void AddChild(ChildItem child)
65          {
66              Children.Add(child);
67              child.Parent = this;
68          }
69
70          public void AddProperty(Property property)
71          {
72              Properties.Add(property);
73              property.Owner = this;
74          }
75
76          public void AddOwnedContract(Contract contract)
77          {
78              OwnedContracts.Add(contract);
79              contract.Owner = this;
80          }
81
82          public void AddRelation(Relation relation)
83          {
84              Relations.Add(relation);
85          }
86      }
87  }
```

```
126 | 1 2 3 relation.Items.Add(this);
127 | | | }
128 | | |
133 | □ public void DeleteRelation(Relation relation)
134 | | {
135 | | Relations.Remove(relation);
136 | | relation.Items.Remove(this);
137 | | }
138 | | }
139 | | }
```

```
14
15  using System;
16  using System.ComponentModel.DataAnnotations;
17  using System.ComponentModel.DataAnnotations.Schema;
18
19  namespace WATB.Datamodel.Generic
20  {
24      [Table("Links")]
25      public class BaseLink : DataContent
26      {
31          public override string ToString()
32          {
33              return string.Format(
34                  "{0}\tLink {7} type {8}:\r\n\towner: {1} ({2})\r\n\tAsset: {3} ({4})
35                  \r\n\tContract: {5} ({6})\r\n", base.ToString(), Master.Name,
36                  Master.Id, Asset.Name, Asset.Id, Contract.Name, Contract.Id,
37                  TimeStamp.ToString("yyyy-MM-dd HH:mm:ss tt zzz"), LinkType);
38          }
39
40          [Required(ErrorMessage = "Time stamp is required.")]
41          public DateTime TimeStamp { get; set; }
42
43          [InverseProperty("OwnedLinks")]
44          [Required(ErrorMessage = "Master is required.")]
45          public virtual BaseItem Master { get; set; }
46
47          [InverseProperty("Links")]
48          [Required(ErrorMessage = "Asset is required.")]
49          public virtual Asset Asset { get; set; }
50
51          [InverseProperty("Transactions")]
52          [Required(ErrorMessage = "Contract is required.")]
53          public virtual BaseContract Contract { get; set; }
54
55          [Required(ErrorMessage = "Need to specify the type of the link.")]
56          public LinkType LinkType { get; set; }
57      }
58  }
```

```
14
15  using System.ComponentModel.DataAnnotations;
16  using System.ComponentModel.DataAnnotations.Schema;
17
18  namespace WATB.Datamodel.Generic
19  {
23      [Table("Children")]
24      public class ChildItem : BaseItem
25      {
30          public override string ToString()
31          {
32              return string.Format("{0}\tParent: {1} ({2})", base.ToString(),
33                                     Parent.Name, Parent.Id);
34          }
35      }
36      [InverseProperty("Children")]
37      [Required(ErrorMessage = "Parent is required.")]
38      public virtual BaseItem Parent { get; set; }
39  }
40
41
42
```

```
14
15  using System.ComponentModel.DataAnnotations;
16  using System.ComponentModel.DataAnnotations.Schema;
17
18  namespace WATB.Datamodel.Generic
19  {
23      [Table("OwnedContracts")]
24      public class Contract : BaseContract
25      {
30          public override string ToString()
31          {
32              return string.Format("{0}\tOwner: {1} ({2})", base.ToString(),
33                                     Owner.Name, Owner.Id);
34          }
39          [InverseProperty("OwnedContracts")]
40          [Required(ErrorMessage = "Owner is required.")]
41          public virtual BaseItem Owner { get; set; }
42      }
43 }
```

```
14
15 namespace WATB.Datamodel.Generic
16 {
20     public enum ContractStatus
21     {
25         InProgress,
26
30         Closed,
31
35         Canceled
36     }
37 }
```

```
14
15  using System.ComponentModel.DataAnnotations;
16  using System.ComponentModel.DataAnnotations.Schema;
17
18  namespace WATB.Datamodel.Generic
19  {
23      [Table("CountLinks")]
24      public class CountLink : BaseLink
25      {
30          public override string ToString()
31          {
32              return string.Format("{0}\tCount: {1}", base.ToString(), Count);
33          }
38          [Required(ErrorMessage = "Need to specify the number of subject items.")
39          ]
39          public int Count { get; set; }
40      }
41  }
```

```
14
15  using System.ComponentModel.DataAnnotations.Schema;
16  using System.Xml.Linq;
17
18  namespace WATB.Datamodel.Generic
19  {
23      [Table("Contents")]
24      public class DataContent : Identifier
25      {
30          public override string ToString()
31          {
32              return string.Format(
33                  "{0}\tContent {1} template\r\n\tContent: {2}\r\n", base.ToString(),
34                  Template == null ? "without" : "with", Content);
35          }
36
37          private XElement _content;
38
43          public string ContentString { get; set; }
44
49          public DataTemplate Template { get; set; }
50
55          [NotMapped]
56          public XElement Content
57          {
58              get
59              {
60                  if (_content == null)
61                  {
62                      _content = string.IsNullOrEmpty(ContentString) ? new
63                          XElement(GetType().Name) : XElement.Parse(ContentString);
64                  }
65                  _content.Changed += _content_Changed;
66                  return _content;
67              }
68              set
69              {
70                  _content = value ?? new XElement(GetType().Name);
71                  _content.Changed += _content_Changed;
72                  ContentString = _content.ToString();
73              }
74          }
80          private void _content_Changed(object sender, XObjectChangeEventArgs e) {
81              ContentString = _content.ToString(); }
86          public bool Validate() { return Template == null || Template.Validate(
87              Content); }
88      }

```

```
14
15  using System.Collections.Generic;
16  using System.ComponentModel.DataAnnotations.Schema;
17  using System.Xml.Linq;
18
19  namespace WATB.Datamodel.Generic
20  {
24      [Table("Templates")]
25      public class DataTemplate : Identifier
26      {
31          public override string ToString()
32          {
33              return string.Format(
34                  "{0}\tSchema:\r\n\t{1}\r\n\tStylesheet:\r\n\t{2}\r\n", base.
35                  ToString(), Schema, Stylesheet);
36          }
37
39          private XElement _schema;
40
44          private XElement _stylesheet;
45
50          public virtual ICollection<DataContent> Contents { get; set; }
51
56          public string SchemaString { get; set; }
57
62          public string StylesheetString { get; set; }
63
68          [NotMapped]
69          public XElement Schema
70          {
71              get
72              {
73                  if (_schema == null)
74                  {
75                      _schema = string.IsNullOrEmpty(SchemaString) ? new
76                          XElement(GetType().Name) : XElement.Parse(SchemaString);
77                      _schema.Changed += _schema_Changed;
78                  }
79                  return _schema;
80              }
81              set
82              {
83                  _schema = value ?? new XElement(GetType().Name);
84                  _schema.Changed += _schema_Changed;
85                  SchemaString = _schema.ToString();
86              }
87          }
92          [NotMapped]
93          public XElement Stylesheet
94          {
95              get
96              {
97                  if (_stylesheet == null)
98                  {
99                      _stylesheet = string.IsNullOrEmpty(StylesheetString) ?
100                          new XElement(GetType().Name) : XElement.Parse(
101                              StylesheetString);
102                      _stylesheet.Changed += _stylesheet_Changed;
103                  }
104                  return _stylesheet;
105              }
106              set
107              {

```

```
106 | 1      2      3      4      _stylesheet = value ?? new XElement(GetType().Name);
107 |      _stylesheet.Changed += _stylesheet_Changed;
108 |      StylesheetString = _stylesheet.ToString();
109 |      }
110 |    }
111 |
117 | private void _schema_Changed(object sender, XObjectChangeEventArgs e) {
118 |     SchemaString = _schema.ToString(); }
124 | private void _stylesheet_Changed(object sender, XObjectChangeEventArgs e
125 | ) { StylesheetString = _stylesheet.ToString(); }
131 | public bool Validate(XElement content) { return (Schema == null ||
132 | ValidateSchema(content)) & (Stylesheet == null || ValidateStylesheet(
133 | content)); }
139 | public bool ValidateSchema(XElement content) { return true; }
140 |
147 | public bool ValidateStylesheet(XElement content) { return true; }
148 | }
149 | }
```

```
14
15  using System;
16  using System.ComponentModel.DataAnnotations;
17  using System.ComponentModel.DataAnnotations.Schema;
18
19  namespace WATB.Datamodel.Generic
20  {
24      [Table("Identifiers")]
25      public class Identifier
26      {
31          public override string ToString()
32          {
33              return string.Format("Type: {0}, ID: {1}\r\n", GetType().Name, Id);
34          }
39          [Required(ErrorMessage = "Identifier is required.")]
40          public Guid Id { get; set; }
41      }
42  }
```

```
14
15 namespace WATB.Datamodel.Generic
16 {
20     public enum LinkType
21     {
25         Link,
26
30         Send,
31
35         Receive
36     }
37 }
```

```
14
15 using System.ComponentModel.DataAnnotations.Schema;
16
17 namespace WATB.Datamodel.Generic
18 {
22     [Table("Properties")]
23     public class Property : Relation
24     {
29         public override string ToString()
30         {
31             return string.Format("{0}\r\n\tOwner: {1} ({2})", base.ToString(),
32                 Owner.Name, Owner.Id);
33         }
38         [InverseProperty("Properties")]
39         public BaseItem Owner { get; set; }
40     }
41 }
```

```
14
15  using System;
16  using System.Collections.Generic;
17  using System.ComponentModel.DataAnnotations.Schema;
18
19  namespace WATB.Datamodel.Generic
20  {
24      [Table("Relations")]
25      public class Relation : BaseItem
26      {
31          public override string ToString()
32          {
33              return string.Format("{0}\t{1} items", base.ToString(), Items.Count);
34          }
35
40          [InverseProperty("Relations")]
41          public virtual ICollection<BaseItem> Items { get; set; }
42
50          public void Add(BaseItem item)
51          {
52              if (item == this) { throw new Exception(
53                  "Relation cannot be added to itself."); }
54              if (Items.Contains(this)) { throw new Exception(
55                  "Item already exists."); }
56              Items.Add(item);
57              item.Relations.Add(this);
63          public void Delete(BaseItem item)
64          {
65              if (!Items.Contains(this)) { throw new Exception(
66                  "Item does not exists."); }
67              Items.Remove(item);
68              item.Relations.Remove(this);
69          }
70      }
}
```

```
1  using System;
2  using System.IO;
3  using System.Linq;
4  using Microsoft.VisualStudio.TestTools.UnitTesting;
5  using WATB.Datamodel.Context;
6  using WATB.Datamodel.Generic;
7
8  namespace WATB.Datamodel.Testing
9  {
13     [TestClass]
14     public sealed class UnitTestContract : IDisposable
15     {
19         private const string DatabaseModel = "ContractModel";
20
25         public GenericDatamodel Db { get; private set; }
26
30         public void Dispose()
31         {
32             if (Db == null) { return; }
33             Db.Dispose();
34             Db = null;
35         }
36
40         [TestInitialize]
41         public void Initialize() { Db = new GenericDatamodel(DatabaseModel); }
42
46         [TestCleanup]
47         public void WhatIsInTheDatabase()
48         {
49             using (var database = new GenericDatamodel(DatabaseModel))
50             {
51                 var idList = database.Identifiers.ToList();
52                 var path = AppDomain.CurrentDomain.BaseDirectory;
53                 using (var output = new StreamWriter(Path.Combine(path,
54                                     DatabaseModel + ".txt"), true))
55                 {
56                     output.WriteLine(
57                         "{0}\r\nNumber of records: {1}\r\nTime: {2}\r\n{0}", new
58                         string('-', 80), idList.Count, DateTime.Now.ToString(
59                         "yyyy-MM-dd HH:mm:ss tt zzz"));
60                     foreach (var identifier in idList)
61                     {
62                         try { output.WriteLine(identifier); }
63                         catch (Exception ex)
64                         {
65                             output.WriteLine("\t{0} ({1})\r\n{2}",
66                                 identifier.GetType().Name, identifier.Id, ex.
67                                 Message);
68                         }
69                         output.WriteLine(new string('-', 80));
70                     }
71                     output.Close();
72                 }
73             }
74
75         [ClassInitialize]
76         public static void ClassInit(TestContext context)
77         {
78             using (var db = new GenericDatamodel(DatabaseModel))
79             {
80                 if (db.Database.Exists()) { db.Database.Delete(); }
81                 db.Database.Create();
82             }
83         }
84     }
85 }
```

```
82 |         }
83 |
84 |     [TestMethod]
85 |     public void TestMakeContract()
86 |     {
87 |         BaseItem owner = Db.AddItem("Owner");
88 |         BaseItem recipient = Db.AddItem("Recipient");
89 |         Asset asset = Db.AddAsset("Product");
90 |         Db.SaveChanges();
91 |
92 |         BaseContract baseContract = Db.AddBaseContract("Contract",
93 |             ContractStatus.InProgress);
94 |         Assert.IsNotNull(baseContract, "Failed to create contract.");
95 |         Assert.AreEqual(baseContract.Status, ContractStatus.InProgress,
96 |             "Invalid status for contract.");
97 |         Assert.AreEqual(baseContract.Transactions.Count, 0,
98 |             "Invalid number of transactions.");
99 |         Db.SaveChanges();
100 |
101 |         BaseLink send = Db.AddSentLink(baseContract, owner, asset);
102 |         Assert.IsNotNull(send, "Failed to create Send link.");
103 |         Assert.AreEqual(send.Contract, baseContract,
104 |             "Send link has the wrong contract.");
105 |         Assert.AreEqual(send.Master, owner, "Link has the wrong owner.");
106 |         Assert.AreEqual(send.Asset, asset, "Link has the wrong product.");
107 |         Assert.AreEqual(send.LinkType, LinkType.Send,
108 |             "Link is of the wrong type.");
109 |         Assert.AreEqual(baseContract.Transactions.Count, 1,
110 |             "Invalid number of transactions.");
111 |         Assert.AreEqual(owner.OwnedLinks.Count, 1,
112 |             "Owner doesn't have a product.");
113 |         Assert.AreEqual(asset.Links.Count, 1,
114 |             "Product doesn't have an owner.");
115 |         Db.SaveChanges();
116 |
117 |         BaseLink receive = Db.AddReceivedLink(baseContract, recipient,
118 |             asset);
119 |         Assert.IsNotNull(receive, "Failed to create Receive link.");
120 |         Assert.AreEqual(receive.Contract, baseContract,
121 |             "Receive link has the wrong contract.");
122 |         Assert.AreEqual(receive.Master, recipient,
123 |             "Link has the wrong owner.");
124 |         Assert.AreEqual(receive.Asset, asset, "Link has the wrong product."
125 |             );
126 |         Assert.AreEqual(receive.LinkType, LinkType.Receive,
127 |             "Link is of the wrong type.");
128 |         Assert.AreEqual(baseContract.Transactions.Count, 2,
129 |             "Invalid number of transactions.");
130 |         Assert.AreEqual(asset.Links.Count, 2,
131 |             "Product doesn't have a recipient.");
132 |         Assert.AreEqual(recipient.OwnedLinks.Count, 1,
133 |             "Owner doesn't have a product.");
134 |         Db.SaveChanges();
135 |     }
136 |
137 |     [TestMethod]
138 |     public void TestMakeOwnedContract()
139 |     {
140 |         BaseItem bank = Db.AddItem("Bank");
141 |         BaseItem owner = Db.AddItem("Owner");
142 |         BaseItem recipient = Db.AddItem("Recipient");
143 |         Asset asset = Db.AddAsset("Product");
144 |         Db.SaveChanges();
145 |     }
146 | }
```

```
1      2      3
136      Contract baseContract = Db.AddContract("Contract", bank,
137      ContractStatus.InProgress);
138      Assert.IsNotNull(baseContract, "Failed to create contract.");
139      Assert.AreEqual(baseContract.Owner, bank, "Failed to assign owner."
140      );
141      Assert.AreEqual(baseContract.Status, ContractStatus.InProgress,
142      "Invalid status for contract.");
143      Assert.AreEqual(baseContract.Transactions.Count, 0,
144      "Invalid number of transactions.");
145      Db.SaveChanges();
146      BaseLink send = Db.AddSentLink(baseContract, owner, asset);
147      Assert.IsNotNull(send, "Failed to create Send link.");
148      Assert.AreEqual(send.Contract, baseContract,
149      "Send link has the wrong contract.");
150      Assert.AreEqual(send.Master, owner, "Link has the wrong owner.");
151      Assert.AreEqual(send.Asset, asset, "Link has the wrong product.");
152      Assert.AreEqual(send.LinkType, LinkType.Send,
153      "Link is of the wrong type.");
154      Assert.AreEqual(baseContract.Transactions.Count, 1,
155      "Invalid number of transactions.");
156      Assert.AreEqual(owner.OwnedLinks.Count, 1,
157      "Owner doesn't have a product.");
158      Assert.AreEqual(asset.Links.Count, 1,
159      "Product doesn't have an owner.");
160      Db.SaveChanges();
161      BaseLink receive = Db.AddReceivedLink(baseContract, recipient,
162      asset);
163      Assert.IsNotNull(receive, "Failed to create Receive link.");
164      Assert.AreEqual(receive.Contract, baseContract,
165      "Receive link has the wrong contract.");
166      Assert.AreEqual(receive.Master, recipient,
167      "Link has the wrong owner.");
168      Assert.AreEqual(receive.Asset, asset, "Link has the wrong product."
169      );
170      Assert.AreEqual(receive.LinkType, LinkType.Receive,
171      "Link is of the wrong type.");
172      Assert.AreEqual(baseContract.Transactions.Count, 2,
173      "Invalid number of transactions.");
174      Assert.AreEqual(asset.Links.Count, 2,
175      "Product doesn't have a recipient.");
176      Assert.AreEqual(recipient.OwnedLinks.Count, 1,
177      "Owner doesn't have a product.");
178      Db.SaveChanges();
179  }
180  }
181  }
```

```
1  using System;
2  using System.IO;
3  using System.Linq;
4  using System.Xml.Linq;
5  using Microsoft.VisualStudio.TestTools.UnitTesting;
6  using WATB.Datamodel.Context;
7  using WATB.Datamodel.Generic;
8
9  namespace WATB.Datamodel.Testing
10 {
14     [TestClass]
15     public sealed class UnitTestItemFactory : IDisposable
16     {
20         private const string DatabaseModel = "ItemModel";
21
26         public GenericDatamodel Db { get; private set; }
27
31         public void Dispose()
32         {
33             if (Db == null) { return; }
34             Db.Dispose();
35             Db = null;
36         }
37
41         [TestInitialize]
42         public void Initialize() { Db = new GenericDatamodel(DatabaseModel); }
43
47         [TestCleanup]
48         public void WhatIsInTheDatabase()
49         {
50             using (var database = new GenericDatamodel(DatabaseModel))
51             {
52                 var idList = database.Identifiers.ToList();
53                 var path = AppDomain.CurrentDomain.BaseDirectory;
54                 using (var output = new StreamWriter(Path.Combine(path, DatabaseModel + ".txt"), true))
55                 {
56                     output.WriteLine(
57                         "{0}\r\nNumber of records: {1}\r\nTime: {2}\r\n{0}", new
58                         string('-', 80), idList.Count, DateTime.Now.ToString(
59                         "yyyy-MM-dd HH:mm:ss tt zzz"));
60                     foreach (var identifier in idList)
61                     {
62                         try { output.WriteLine(identifier); }
63                         catch (Exception ex)
64                         {
65                             output.WriteLine("\t{0} ({1})\r\n{2}",
66                                 identifier.GetType().Name, identifier.Id, ex.
67                                 Message);
68                         }
69                     }
70                     output.WriteLine(new string('-', 80));
71                 }
72                 output.Close();
73             }
74         }
75
76         [ClassInitialize]
77         public static void ClassInit(TestContext context)
78         {
79             using (var db = new GenericDatamodel(DatabaseModel))
80             {
81                 if (db.Database.Exists()) { db.Database.Delete(); }
82                 db.Database.Create();
83             }
84         }
85     }
86 }
```

```
82 |         1     2     3     4
83 |         |         |         |         |
84 |         |         |         |         |
88 |         [TestMethod]
89 |         public void TestMakeTemplate()
90 |         {
91 |             DataTemplate template = Db.AddTemplate(new XElement("schema"), new XElement("Stylesheet"));
92 |             Assert.IsNotNull(template, "Failed to create template.");
93 |             Assert.IsNotNull(template.Schema, "Failed to create template schema.");
94 |             Assert.IsNotNull(template.Stylesheet, "Failed to create template stylesheet.");
95 |             Db.SaveChanges();
96 |         }
97 |
101 |        [TestMethod]
102 |        public void TestMakeContent()
103 |        {
104 |            DataTemplate template = Db.AddTemplate(new XElement("schema"), new XElement("Stylesheet"));
105 |            DataContent content = Db.AddContent(new XElement("Hello"), template);
106 |            Assert.IsNotNull(content, "Failed to create content.");
107 |            Assert.IsNotNull(content.Content, "Failed to create content data.");
108 |            Assert.IsNotNull(content.Template, "Failed to create template.");
109 |            Db.SaveChanges();
110 |        }
111 |
115 |        [TestMethod]
116 |        public void TestMakeItem()
117 |        {
118 |            BaseItem item = Db.AddItem("Item");
119 |            Assert.IsNotNull(item, "Failed to create item.");
120 |            Db.SaveChanges();
121 |        }
122 |
126 |        [TestMethod]
127 |        public void TestMakeChild()
128 |        {
129 |            BaseItem parent = Db.AddItem("Parent");
130 |            ChildItem child = Db.AddChild("Child", parent);
131 |            Assert.IsNotNull(child, "Failed to create child.");
132 |            Assert.AreEqual(child.Parent, parent, "Child has the wrong parent.");
133 |            Assert.AreEqual(parent.Children.Count, 1, "Parent doesn't have the proper amount of children.");
134 |            Db.SaveChanges();
135 |        }
136 |
140 |        [TestMethod]
141 |        public void TestMakeEmptyRelation()
142 |        {
143 |            BaseItem item = Db.AddItem("Item");
144 |            Relation relation = Db.AddRelation("Relation");
145 |            Assert.IsNotNull(relation, "Failed to create empty relation.");
146 |            relation.Add(item);
147 |            Assert.AreEqual(relation.Items.Count, 1, "Relation doesn't have the proper amount of items.");
148 |            Assert.AreEqual(item.Relations.Count, 1, "Related item doesn't have the proper amount of relations.");
149 |            Db.SaveChanges();
150 |        }
```

```
151 | 1 2
155 | [TestMethod]
156 | public void TestMakeFilledRelation()
157 | {
158 |     BaseItem item1 = Db.AddItem("Item 1");
159 |     Assert.IsNotNull(item1, "Failed to create item 1 for relation.");
160 |     BaseItem item2 = Db.AddItem("Item");
161 |     Assert.IsNotNull(item2, "Failed to create item 2 for relation.");
162 |     Relation relation = Db.AddRelation("Relation", null, null, item1,
163 |     item2);
164 |     Assert.IsNotNull(relation, "Failed to create empty relation.");
165 |     Assert.AreEqual(relation.Items.Count, 2,
166 |     "Relation doesn't have the proper amount (2) of items.");
167 |     Assert.AreEqual(item1.Relations.Count, 1,
168 |     "Related item 1 doesn't have the proper amount of relations.");
169 |     Assert.AreEqual(item2.Relations.Count, 1,
170 |     "Related item 2 doesn't have the proper amount of relations.");
171 |     Db.SaveChanges();
172 | }
173 | [TestMethod]
174 | public void TestDeleteRelation()
175 | {
176 |     BaseItem item1 = Db.AddItem("Item 1");
177 |     BaseItem item2 = Db.AddItem("Item");
178 |     Relation relation = Db.AddRelation("Relation", null, null, item1,
179 |     item2);
180 |     Db.SaveChanges();
181 |     item1.DeleteRelation(relation);
182 |     Assert.AreEqual(relation.Items.Count, 1,
183 |     "Relation doesn't have the proper amount (1) of items.");
184 |     Assert.AreEqual(item1.Relations.Count, 0,
185 |     "Related item 1 doesn't have the proper amount of relations.");
186 |     Assert.AreEqual(item2.Relations.Count, 1,
187 |     "Related item 2 doesn't have the proper amount of relations.");
188 |     Db.SaveChanges();
189 | }
190 | [TestMethod]
191 | public void TestMakeEmptyProperty()
192 | {
193 |     BaseItem owner = Db.AddItem("Owner");
194 |     BaseItem item = Db.AddItem("Item");
195 |     Property property = Db.AddProperty("Property", owner);
196 |     Assert.IsNotNull(property, "Failed to create empty relation.");
197 |     Assert.AreEqual(property.Owner, owner, "Failed to assign owner.");
198 |     item.AddRelation(property);
199 |     Assert.AreEqual(owner.Properties.Count, 1,
200 |     "Owner doesn't have the proper amount of properties.");
201 |     Assert.AreEqual(property.Items.Count, 1,
202 |     "Property doesn't have the proper amount of relations.");
203 |     Assert.AreEqual(item.Relations.Count, 1,
204 |     "Related item doesn't have the proper amount of relations.");
205 |     Db.SaveChanges();
206 | }
207 | [TestMethod]
208 | public void TestMakeFilledProperty()
209 | {
210 |     BaseItem owner = Db.AddItem("Owner");
211 |     BaseItem item1 = Db.AddItem("Item 1");
212 |     BaseItem item2 = Db.AddItem("Item");
213 |     Property property = Db.AddProperty("Property", owner, null, null,
214 |     item1, item2);
```

```
1      2      3
215 |         Assert.IsNotNull(property, "Failed to create empty relation.");
216 |         Assert.AreEqual(property.Owner, owner, "Failed to assign owner.");
217 |         Assert.AreEqual(property.Items.Count, 2,
218 |             "Relation doesn't have the proper amount (2) of items.");
219 |         Assert.AreEqual(item1.Relations.Count, 1,
220 |             "Related item 1 doesn't have the proper amount of relations.");
221 |         Assert.AreEqual(item2.Relations.Count, 1,
222 |             "Related item 2 doesn't have the proper amount of relations.");
223 |         Db.SaveChanges();
224 |     }
225 |
226 |     [TestMethod]
227 |     public void TestDeleteProperty()
228 |     {
229 |         BaseItem owner = Db.AddItem("Owner");
230 |         BaseItem item1 = Db.AddItem("Item 1");
231 |         BaseItem item2 = Db.AddItem("Item");
232 |         Property property = Db.AddProperty("Property", owner, null, null,
233 |             item1, item2);
234 |         Db.SaveChanges();
235 |         item1.DeleteRelation(property);
236 |         Assert.AreEqual(property.Items.Count, 1,
237 |             "Relation doesn't have the proper amount (1) of items.");
238 |         Assert.AreEqual(item1.Relations.Count, 0,
239 |             "Related item 1 doesn't have the proper amount of relations.");
240 |         Assert.AreEqual(item2.Relations.Count, 1,
241 |             "Related item 2 doesn't have the proper amount of relations.");
242 |         Db.SaveChanges();
243 |     }
244 |
245 |     [TestMethod]
246 |     public void TestMakeSubject()
247 |     {
248 |         Asset subject = Db.AddAsset("Subject");
249 |         Assert.IsNotNull(subject, "Failed to create subject.");
250 |         Db.SaveChanges();
251 |     }
252 |
253 |     [TestMethod]
254 |     public void TestMakeEmptyContract()
255 |     {
256 |         BaseContract baseContract = Db.AddBaseContract("Contract",
257 |             ContractStatus.InProgress);
258 |         Assert.IsNotNull(baseContract, "Failed to create contract.");
259 |         Assert.AreEqual(baseContract.Status, ContractStatus.InProgress,
260 |             "Invalid status for contract.");
261 |         Db.SaveChanges();
262 |     }
263 |
264 |     [TestMethod]
265 |     public void TestMakeEmptyOwnedContract()
266 |     {
267 |         BaseItem bank = Db.AddItem("Bank");
268 |         Contract baseContract = Db.AddContract("Contract", bank,
269 |             ContractStatus.InProgress);
270 |         Assert.IsNotNull(baseContract, "Failed to create contract.");
271 |         Assert.AreEqual(baseContract.Owner, bank, "Failed to assign owner.");
272 |         Assert.AreEqual(baseContract.Status, ContractStatus.InProgress,
273 |             "Invalid status for contract.");
274 |         Db.SaveChanges();
275 |     }
276 | }
277 |
278 | }
```

279

1
}

```
1  using System;
2  using System.IO;
3  using System.Linq;
4  using Microsoft.VisualStudio.TestTools.UnitTesting;
5  using WATB.Datamodel.Context;
6  using WATB.Datamodel.Generic;
7
8  namespace WATB.Datamodel.Testing
9  {
13     [TestClass]
14     public sealed class UnitTestLinkItems : IDisposable
15     {
19         private const string DatabaseModel = "LinkModel";
20
25         public GenericDatamodel Db { get; private set; }
26
30         public void Dispose()
31         {
32             if (Db == null) { return; }
33             Db.Dispose();
34             Db = null;
35         }
36
40         [TestInitialize]
41         public void Initialize() { Db = new GenericDatamodel(DatabaseModel); }
42
46         [TestCleanup]
47         public void WhatIsInTheDatabase()
48         {
49             using (var database = new GenericDatamodel(DatabaseModel))
50             {
51                 var idList = database.Identifiers.ToList();
52                 var path = AppDomain.CurrentDomain.BaseDirectory;
53                 using (var output = new StreamWriter(Path.Combine(path,
54                                     DatabaseModel + ".txt"), true))
55                 {
56                     output.WriteLine(
57                         "{0}\r\nNumber of records: {1}\r\nTime: {2}\r\n{0}", new
58                         string('-', 80), idList.Count, DateTime.Now.ToString(
59                         "yyyy-MM-dd HH:mm:ss tt zzz"));
60                     foreach (var identifier in idList)
61                     {
62                         try { output.WriteLine(identifier); }
63                         catch (Exception ex)
64                         {
65                             output.WriteLine("\t{0} ({1})\r\n{2}",
66                                 identifier.GetType().Name, identifier.Id, ex.
67                                 Message);
68                         }
69                         output.WriteLine(new string('-', 80));
70                     }
71                     output.Close();
72                 }
73             }
74
75         [ClassInitialize]
76         public static void ClassInit(TestContext context)
77         {
78             using (var db = new GenericDatamodel(DatabaseModel))
79             {
80                 if (db.Database.Exists()) { db.Database.Delete(); }
81                 db.Database.Create();
82             }
83         }
84     }
85 }
```

```
82 |         }
83 |
87 |         [TestMethod]
88 |         public void TestMakeLink()
89 |         {
90 |             BaseItem owner = Db.AddItem("Owner");
91 |             Asset product = Db.AddAsset("Asset");
92 |             BaseContract baseContract = Db.AddContract("Contract", owner,
93 |             ContractStatus.InProgress);
94 |             int count = 0;
95 |             foreach (LinkType linkType in Enum.GetValues(typeof(LinkType)))
96 |             {
97 |                 count++;
98 |                 BaseLink link = Db.AddLink(linkType, baseContract, owner,
99 |                 product);
100 |                 Assert.IsNotNull(link, string.Format(
101 |                 "Failed to create link for type {0}.", linkType));
102 |                 Assert.AreEqual(link.Master, owner,
103 |                 "Link has the wrong owner.");
104 |                 Assert.AreEqual(link.Asset, product,
105 |                 "Link has the wrong product.");
106 |                 Assert.AreEqual(link.LinkType, linkType,
107 |                 "Link is of the wrong type.");
108 |                 Assert.AreEqual(owner.OwnedLinks.Count, count,
109 |                 "Owner doesn't have a product.");
110 |                 Assert.AreEqual(product.Links.Count, count,
111 |                 "Product doesn't have an owner.");
112 |                 Db.SaveChanges();
113 |             }
114 |         }
115 |
116 |         [TestMethod]
117 |         public void TestMakeIntegerLink()
118 |         {
119 |             BaseItem owner = Db.AddItem("Owner, number");
120 |             Asset product = Db.AddAsset("Asset, number");
121 |             BaseContract baseContract = Db.AddContract("Contract, number",
122 |             owner, ContractStatus.InProgress);
123 |             int count = 0;
124 |             foreach (LinkType linkType in Enum.GetValues(typeof(LinkType)))
125 |             {
126 |                 count++;
127 |                 BaseLink link = Db.AddLink(linkType, baseContract, owner,
128 |                 product, 10);
129 |                 Assert.IsNotNull(link, string.Format(
130 |                 "Failed to create count link for type {0}.", linkType));
131 |                 Assert.IsInstanceOfType(link, typeof(CountLink),
132 |                 "Link is of the wrong type.");
133 |                 Assert.AreEqual(link.Master, owner,
134 |                 "Link has the wrong owner.");
135 |                 Assert.AreEqual(link.Asset, product,
136 |                 "Link has the wrong product.");
137 |                 Assert.AreEqual(link.LinkType, linkType,
138 |                 "Link is of the wrong type.");
139 |                 Assert.AreEqual(owner.OwnedLinks.Count, count,
140 |                 "Owner doesn't have a product.");
141 |                 Assert.AreEqual(product.Links.Count, count,
142 |                 "Product doesn't have an owner.");
143 |                 Db.SaveChanges();
144 |             }
145 |         }
146 |
147 |         [TestMethod]
148 |         public void TestMakeFloatLink()
149 |         {
```

```
138 | 1 2 | {
139 | | BaseItem owner = Db.AddItem("Owner, amount");
140 | | Asset product = Db.AddAsset("Asset, amount");
141 | | BaseContract baseContract = Db.AddContract("Contract, amount",
142 | | owner, ContractStatus.InProgress);
143 | | int count = 0;
144 | | foreach (LinkType linkType in Enum.GetValues(typeof(LinkType)))
145 | | {
146 | |     BaseLink link = Db.AddLink(linkType, baseContract, owner,
147 | |     product, Math.PI);
148 | |     count++;
149 | |     Assert.IsNotNull(link, string.Format(
150 | |     "Failed to create amount link for type {0}.", linkType));
151 | |     Assert.IsInstanceOfType(link, typeof(AmountLink),
152 | |     "Link is of the wrong type.");
153 | |     Assert.AreEqual(link.Master, owner,
154 | |     "Link has the wrong owner.");
155 | |     Assert.AreEqual(link.Asset, product,
156 | |     "Link has the wrong product.");
157 | |     Assert.AreEqual(link.LinkType, linkType,
158 | |     "Link is of the wrong type.");
159 | |     Assert.AreEqual(owner.OwnedLinks.Count, count,
160 | |     "Owner doesn't have a product.");
161 | |     Assert.AreEqual(product.Links.Count, count,
162 | |     "Product doesn't have an owner.");
163 | |     Db.SaveChanges();
164 | | }
165 | | }
166 | | }
167 | | }
168 | | }
```

-
- _content, 20
- _content_Changed, 20
- _schema, 21, 22
- _schema_Changed, 21, 22
- _stylesheet, 21, 22
- _stylesheet_Changed, 21, 22
- A**
- Add, 4-6, 11-14, 26, 31
- AddAsset, 5, 28, 33, 36, 37
- AddBaseContract, 5, 28, 33
- AddChild, 4, 7, 13, 31
- AddContent, 4, 31
- AddContract, 5, 29, 33, 36, 37
- AddItem, 4, 28, 31-33, 36, 37
- AddLink, 5, 36, 37
- AddOwnedContract, 8, 13
- AddOwnerLink, 8, 11
- AddProductLink, 8, 13
- AddProperty, 4, 8, 13, 32, 33
- AddReceivedLink, 5, 28, 29
- AddRelation, 4, 8, 13, 31, 32
- AddSentLink, 5, 28, 29
- AddTemplate, 4, 31
- AddTransaction, 8, 12
- amount, 5, 6, 8, 9
- Amount, 9, 10
- AmountLink, 4, 8, 10, 37
- AmountLinks, 4-6
- AppDomain, 27, 30, 35
- AreEqual, 28, 29, 31-33, 36, 37
- Assert, 28, 29, 31-33, 36, 37
- Asset, 4, 5, 8, 9, 11, 12, 15, 28, 29, 33, 36, 37
- asset, 5, 6, 28, 29
- Assets, 4, 5
- B**
- bank, 28, 29, 33
- BaseContract, 4, 5, 8, 9, 12, 15, 17, 28, 33, 36, 37
- baseContract, 5, 6, 8, 9, 28, 29, 33, 36, 37
- BaseDirectory, 27, 30, 35
- Baseltem, 4, 5, 7-9, 11, 13, 15-17, 25, 26, 28, 31-33, 36, 37
- BaseLink, 4, 5, 7, 8, 10-13, 15, 19, 28, 29, 36, 37
- C**
- Canceled, 18
- Changed, 20-22
- child, 13, 31
- ChildItem, 4, 7, 13, 16, 31
- Children, 4, 7, 13, 31
- ClassNit, 27, 30, 35
- ClassInitialize, 27, 30, 35
- Close, 27, 30, 35
- Closed, 18
- Collections, 7, 11-13, 21, 26
- Combine, 27, 30, 35
- ComponentModel, 10-13, 15-17, 19-21, 23, 25, 26
- Contains, 26
- Content, 7, 20, 31
- content, 4-9, 22, 31
- Contents, 4, 7, 21
- ContentString, 20
- context, 27, 30, 35
- Context, 4, 27, 30, 35
- contract, 8, 13
- Contract, 4, 5, 7, 8, 12, 13, 15, 17, 28, 29, 33
- Contracts, 4, 5
- contracts, 5, 8
- ContractStatus, 5, 8, 12, 18, 28, 29, 33, 36, 37
- Count, 9, 11-13, 19, 26-33, 35-37
- count, 5, 9, 36, 37
- CountLink, 4, 9, 19, 36
- CountLinks, 4, 5
- Create, 4-6, 27, 30, 35
- CurrentDomain, 27, 30, 35
- D**
- Data, 4
- DataAnnotations, 10-13, 15-17, 19-21, 23, 25, 26
- Database, 27, 30, 35
- database, 27, 30, 35
- DatabaseModel, 27, 30, 35
- DataContent, 4, 7, 13, 15, 20, 21, 31
- datamodel, 4
- Datamodel, 4, 7, 10-13, 15-21, 23-27, 30, 35
- DataTemplate, 4, 5, 7-9, 20, 21, 31
- DateTime, 8, 15, 27, 30, 35
- db, 27, 30, 35
- Db, 27-33, 35-37
- DbContext, 4
- DbSet, 4
- Delete, 26, 27, 30, 35
- DeleteRelation, 14, 32, 33
- Dispose, 27, 30, 35
- E**
- e, 20, 22
- Entity, 4
- ErrorMessage, 10, 13, 15-17, 19, 23
- ex, 27, 30, 35
- Exception, 26, 27, 30, 35
- Exists, 27, 30, 35
- F**
- Factory, 4, 7
- Format, 10-13, 15-17, 19-21, 23, 25, 26, 36, 37
- G**
- Generic, 4, 7, 10-13, 15-21, 23-27, 30, 35
- GenericDatamodel, 4, 27, 30, 35
- GenericFactory, 7
- GetType, 20-23, 27, 30, 35
- GetValues, 36, 37
- Guid, 7, 8, 23
- H**
- HashSet, 7, 8
- I**
- i, 8
- ICollection, 11-13, 21, 26
- Id, 7, 8, 15-17, 23, 25, 27, 30, 35
- Identifier, 4, 7, 20, 21, 23
- identifier, 27, 30, 35
- Identifiers, 4, 27, 30, 35
- IDisposable, 27, 30, 35
- idList, 27, 30, 35
- Initialize, 4-9, 27, 30, 35
- InProgress, 18, 28, 29, 33, 36, 37
- InverseProperty, 11-13, 15-17, 25, 26
- IO, 27, 30, 35
- IsInstanceOfType, 36, 37
- IsNotNull, 28, 29, 31-33, 36, 37
- IsNullOrEmpty, 20, 21
- item, 7, 8, 26, 31, 32
- item1, 32, 33
- item2, 32, 33
- Items, 4, 8, 14, 26, 31-33
- items, 4, 5, 8
- L**
- link, 8, 9, 11-13, 36, 37
- Link, 24
- Links, 4, 5, 8, 11, 28, 29, 36, 37
- linkType, 5, 8, 9, 36, 37
- LinkType, 5, 6, 8, 9, 15, 24, 28, 29, 36, 37
- Linq, 4, 7, 20, 21, 27, 30, 35
- M**
- Master, 13, 15, 28, 29, 36, 37
- Math, 37
- MaxLength, 13
- Message, 27, 30, 35
- Microsoft, 27, 30, 35
- N**
- name, 4, 5, 7, 8
- Name, 7, 8, 13, 15-17, 20-23, 25, 27, 30, 35
- NewGuid, 7, 8
- NotMapped, 20, 21
- Now, 8, 27, 30, 35
- O**
- OrderBy, 8
- output, 27, 30, 35
- OwnedContracts, 4, 5, 7, 13
- OwnedLinks, 7, 13, 28, 29, 36, 37
- owner, 4-6, 8, 9, 28, 29, 32, 33, 36, 37
- Owner, 13, 17, 25, 29, 32, 33
- P**
- parent, 4, 7, 31
- Parent, 13, 16, 31
- Parse, 20, 21
- Path, 27, 30, 35
- path, 27, 30, 35
- PI, 37
- product, 8, 9, 36, 37
- Properties, 4, 5, 7, 13, 32
- property, 13, 32, 33
- Property, 4, 7, 8, 13, 25, 32, 33
- public, 4, 5, 7-28, 30-33, 35, 36
- R**
- receive, 28, 29
- Receive, 5, 6, 24, 28, 29
- recipient, 28, 29
- relation, 8, 13, 14, 31, 32
- Relation, 4, 7, 13, 14, 25, 26, 31, 32
- relations, 7, 8
- Relations, 4, 7, 13, 14, 26, 31-33
- Remove, 14, 26
- Required, 10, 12, 13, 15-17, 19, 23
- result, 7-9
- S**
- SaveChanges, 28, 29, 31-33, 36, 37
- schema, 4, 7
- Schema, 7, 10-13, 15-17, 19-23, 25, 26, 31
- SchemaString, 21, 22
- Send, 5, 24, 28, 29
- send, 28, 29
- sender, 20, 22
- Status, 8, 12, 28, 29, 33
- status, 5, 8

StreamWriter, [27](#), [30](#), [35](#)
stylesheet, [4](#), [7](#)
Stylesheet, [7](#), [21](#), [22](#), [31](#)
StylesheetString, [21](#), [22](#)
subject, [33](#)
System, [4](#), [7](#), [10-13](#), [15-17](#), [19-21](#), [23](#), [25-27](#),
[30](#), [35](#)

T

T, [7-9](#)
Table, [10-13](#), [15-17](#), [19-21](#), [23](#), [25](#), [26](#)
template, [4-9](#), [31](#)
Template, [7](#), [20](#), [31](#)
Templates, [4](#)
TestClass, [27](#), [30](#), [35](#)
TestCleanup, [27](#), [30](#), [35](#)
TestContext, [27](#), [30](#), [35](#)
TestDeleteProperty, [33](#)
TestDeleteRelation, [32](#)
Testing, [27](#), [30](#), [35](#)
TestInitialize, [27](#), [30](#), [35](#)
TestMakeChild, [31](#)
TestMakeContent, [31](#)
TestMakeContract, [28](#)
TestMakeEmptyContract, [33](#)
TestMakeEmptyOwnedContract, [33](#)
TestMakeEmptyProperty, [32](#)
TestMakeEmptyRelation, [31](#)
TestMakeFilledProperty, [32](#)
TestMakeFilledRelation, [32](#)
TestMakeFloatLink, [36](#)
TestMakeIntegerLink, [36](#)
TestMakeItem, [31](#)
TestMakeLink, [36](#)
TestMakeOwnedContract, [28](#)
TestMakeSubject, [33](#)
TestMakeTemplate, [31](#)
TestMethod, [28](#), [31-33](#), [36](#)
TestTools, [27](#), [30](#), [35](#)
TimeStamp, [8](#), [15](#)
ToList, [8](#), [27](#), [30](#), [35](#)
ToString, [10-13](#), [15-17](#), [19-23](#), [25-27](#), [30](#), [35](#)
Transactions, [8](#), [12](#), [28](#), [29](#)

U

UnitTestContract, [27](#)
UnitTesting, [27](#), [30](#), [35](#)
UnitTestItemFactory, [30](#)
UnitTestLinkItems, [35](#)

V

Validate, [20](#), [22](#)
ValidateSchema, [22](#)
ValidateStylesheet, [22](#)
var, [27](#), [30](#), [35](#)
VisualStudio, [27](#), [30](#), [35](#)

W

WATB, [4](#), [7](#), [10-13](#), [15-21](#), [23-27](#), [30](#), [35](#)
WhatIsInTheDatabase, [27](#), [30](#), [35](#)
where, [7-9](#)
WriteLine, [27](#), [30](#), [35](#)

X

XElement, [4](#), [5](#), [7-9](#), [20-22](#), [31](#)
Xml, [4](#), [7](#), [20](#), [21](#), [30](#)
XObjectChangeEventArgs, [20](#), [22](#)