

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using MusicStore.Logic.Context;
5
6 namespace MusicStore.Logic.Controllers
7 {
8     internal abstract class GenericController<E, I> : ControllerObject, 7
9         IController<I>
10     {
11         where E : Entities.EntityObject, I, Contracts.ICopyable<I>, new()
12         where I : Contracts.IIdentifiable
13     {
14         protected abstract List<E> Set { get; }
15
16         protected GenericController(ContextObject contextObject)
17             : base(contextObject)
18         {
19         }
20         protected GenericController(ControllerObject controllerObject)
21             : base(controllerObject)
22         {
23         }
24
25         public virtual I Create()
26         {
27             return new E();
28         }
29
30         public virtual IEnumerable<I> GetAll()
31         {
32             return Set.Where(i => i.State != Entities.EntityState.Deleted)
33                 .Select(i =>
34                 {
35                     var result = new E();
36
37                     result.CopyProperties(i);
38                     return result;
39                 });
40         }
41         public virtual I GetById(int id)
42         {
43             var result = default(E);
44             var item = Set.SingleOrDefault(i => i.State != 7
45                 Entities.EntityState.Deleted && i.Id == id);
46
47             if (item != null)
48             {
49                 result = new E();
50                 result.CopyProperties(item);
51             }
52             return result;
53         }
54         protected virtual void BeforeInserting(I entity)
55         {
56         }
```

```
55
56     }
57     public virtual I Insert(I entity)
58     {
59         if (entity == null)
60             throw new ArgumentNullException(nameof(entity));
61
62         BeforeInserting(entity);
63         var insertEntity = new E();
64
65         insertEntity.CopyProperties(entity);
66         insertEntity.Id = 0;
67         insertEntity.State = Entities.EntityState.Added;
68         Set.Add(insertEntity);
69         AfterInserted(insertEntity);
70         return insertEntity;
71     }
72     protected virtual void AfterInserted(E entity)
73     {
74
75     }
76
77     protected virtual void BeforeUpdating(I entity)
78     {
79
80     }
81     public virtual void Update(I entity)
82     {
83         if (entity == null)
84             throw new ArgumentNullException(nameof(entity));
85
86         BeforeUpdating(entity);
87         var updateEntity = Set.SingleOrDefault(i => i.State !=
88             Entities.EntityState.Deleted && i.Id == entity.Id);
89
90         if (updateEntity != null)
91         {
92             updateEntity.CopyProperties(entity);
93             AfterUpdated(updateEntity);
94             updateEntity.State = Entities.EntityState.Modified;
95         }
96     }
97     protected virtual void AfterUpdated(E entity)
98     {
99
100     }
101     protected virtual void BeforeDeleting(I entity)
102     {
103
104     }
105     public void Delete(int id)
106     {
107         var item = Set.SingleOrDefault(i => i.State !=
108             Entities.EntityState.Deleted && i.Id == id);
```

```
109         if (item != null)
110         {
111             BeforeDeleting(item);
112             item.State = Entities.EntityState.Deleted;
113             AfterDeleted(item);
114         }
115     }
116     protected virtual void AfterDeleted(E entity)
117     {
118     }
119 }
120
121 public void Save()
122 {
123     Context.Save();
124 }
125 }
126 }
127
```