

```

1  /*-----
2  * Author:      Dan Cassidy
3  * Date:        2015-06-23
4  * Assignment:  cView-P4
5  * Source File: Add.aspx.cs
6  * Language:    C#
7  * Course:      CSCI-C 490, C# Programming, MoWe 08:00
8  * Purpose:     Code-behind file for Add.aspx. Controls the process of adding an item to the
9  *              database via the Entity Framework model.
10 -----*/
11
12 using System;
13 using System.Collections.Generic;
14 using System.Linq;
15 using System.Web;
16 using System.Web.UI;
17 using System.Web.UI.WebControls;
18
19 namespace cView_P4_DanCassidy
20 {
21     public partial class Add : System.Web.UI.Page
22     {
23         /*-----
24         * Name:      btnAdd_Click
25         * Type:      Event Handler Method
26         * Purpose:   Handles constructing, errorchecking, and finally adding an object to the
27         *              database.
28         * Input:     object sender, holds a reference to the object that raised this event.
29         * Input:     EventArgs e, holds data related to this event.
30         * Output:    Nothing.
31         -----*/
32         protected void btnAdd_Click(object sender, EventArgs e)
33         {
34             object toAdd = null;
35
36             switch ((Global.Enums.ItemTypes)ddlItemType.SelectedIndex)
37             {
38                 case Global.Enums.ItemTypes.Business:
39                     toAdd = new Business()
40                     {
41                         // Common Fields
42                         Name = txtName.Text.Trim(),
43                         Type = txtType.Text.Trim(),
44                         StreetAddress = txtStreetAddress.Text.Trim(),
45                         City = txtCity.Text.Trim(),
46                         State = txtState.Text.Trim(),
47                         Zip = txtZip.Text.Trim(),
48                         Latitude = SimpleConvert.ToDecimal(txtLatitude.Text.Trim()),
49                         Longitude = SimpleConvert.ToDecimal(txtLongitude.Text.Trim()),
50                         Phone = txtPhone.Text.Trim(),
51
52                         // Business Fields
53                         LicenseNumber = txtLicenseNumber.Text.Trim(),
54                         LicenseIssueDate = SimpleConvert.ToDateTime(
55                             txtLicenseExpirDate.Text.Trim()),
56                         LicenseExpirDate = SimpleConvert.ToDateTime(
57                             txtLicenseExpirDate.Text.Trim()),
58                         LicenseStatus = txtLicenseStatus.Text.Trim(),
59                         CouncilDistrict = txtCouncilDistrict.Text.Trim()
60                     };
61                     break;
62
63                 case Global.Enums.ItemTypes.Park:
64                     toAdd = new Park()
65                     {
66                         // Common Fields

```

```

67         Name = txtName.Text.Trim(),
68         Type = txtType.Text.Trim(),
69         StreetAddress = txtStreetAddress.Text.Trim(),
70         City = txtCity.Text.Trim(),
71         State = txtState.Text.Trim(),
72         Zip = txtZip.Text.Trim(),
73         Latitude = SimpleConvert.ToDecimal(txtLatitude.Text.Trim()),
74         Longitude = SimpleConvert.ToDecimal(txtLongitude.Text.Trim()),
75         Phone = txtPhone.Text.Trim(),
76
77         // Park Fields
78         FeatureBaseball = SimpleConvert.ToByte(txtFeatureBaseball.Text.Trim()),
79         FeatureBasketball = SimpleConvert.ToDecimal(
80             txtFeatureBasketball.Text.Trim()),
81         FeatureGolf = SimpleConvert.ToDecimal(txtFeatureGolf.Text.Trim()),
82         FeatureLargeMPField = SimpleConvert.ToByte(
83             txtFeatureLargeMPField.Text.Trim()),
84         FeatureTennis = SimpleConvert.ToByte(txtFeatureTennis.Text.Trim()),
85         FeatureVolleyball = SimpleConvert.ToByte(txtFeatureVolleyball.Text.Trim())
86     };
87     break;
88
89     case Global.Enums.ItemTypes.PublicFacility:
90         toAdd = new PublicFacility()
91         {
92             // Common Fields
93             Name = txtName.Text.Trim(),
94             Type = txtType.Text.Trim(),
95             StreetAddress = txtStreetAddress.Text.Trim(),
96             City = txtCity.Text.Trim(),
97             State = txtState.Text.Trim(),
98             Zip = txtZip.Text.Trim(),
99             Latitude = SimpleConvert.ToDecimal(txtLatitude.Text.Trim()),
100             Longitude = SimpleConvert.ToDecimal(txtLongitude.Text.Trim()),
101             Phone = txtPhone.Text.Trim()
102         };
103     break;
104
105     default:
106         // ... How?
107         return;
108 }
109
110 // Add the object to the database.
111 if (toAdd != null)
112 {
113     try
114     {
115         using (CViewDataEntities database = new CViewDataEntities())
116         {
117             if (toAdd is Business)
118             {
119                 Business businessToAdd = toAdd as Business;
120
121                 // Do error-checking.
122                 if (businessToAdd.LicenseNumber == string.Empty ||
123                     businessToAdd.LicenseNumber == null)
124                 {
125                     throw new Global.Exceptions.EmptyOrNullPKException(
126                         Global.Strings.BusinessKey);
127                 }
128                 else if (database.Businesses.Find(businessToAdd.LicenseNumber) != null)
129                 {
130                     throw new Global.Exceptions.DuplicatePKException(
131                         Global.Strings.BusinessKey, businessToAdd.LicenseNumber);
132                 }

```

```

133
134         // If everything is ok, add item to table.
135         database.Businesses.Add(businessToAdd);
136     }
137     else if (toAdd is Park)
138     {
139         Park parkToAdd = toAdd as Park;
140
141         //Do error-checking.
142         if (parkToAdd.Name == string.Empty ||
143             parkToAdd.Name == null)
144         {
145             throw new Global.Exceptions.EmptyOrNullPKException(
146                 Global.Strings.ParkKey);
147         }
148         else if (database.Parks.Find(parkToAdd.Name) != null)
149         {
150             throw new Global.Exceptions.DuplicatePKException(
151                 Global.Strings.ParkKey, parkToAdd.Name);
152         }
153
154         // If everything is ok, add item to table.
155         database.Parks.Add(parkToAdd);
156     }
157     else if (toAdd is PublicFacility)
158     {
159         PublicFacility publicFacilityToAdd = toAdd as PublicFacility;
160
161         //Do error-checking.
162         if (publicFacilityToAdd.Name == string.Empty ||
163             publicFacilityToAdd.Name == null)
164         {
165             throw new Global.Exceptions.EmptyOrNullPKException(
166                 Global.Strings.PublicFacilityKey);
167         }
168         else if (database.PublicFacilities.Find(publicFacilityToAdd.Name) !=
169             null)
170         {
171             throw new Global.Exceptions.DuplicatePKException(
172                 Global.Strings.PublicFacilityKey, publicFacilityToAdd.Name);
173         }
174
175         // If everything is ok, add item to table.
176         database.PublicFacilities.Add(publicFacilityToAdd);
177     }
178
179     database.SaveChanges();
180
181     lblResult.Text = "Added record to the table.";
182     lblResult.Visible = true;
183     lblError.Visible = false;
184 }
185 }
186 catch (Exception ex)
187 {
188     // Drill down to the innermost exception.
189     while (ex.InnerException != null)
190         ex = ex.InnerException;
191
192     lblError.Text = "Error: " + ex.Message;
193     lblError.Visible = true;
194     lblResult.Visible = false;
195 }
196 }
197 }
198

```

```

199  /*-----
200  * Name:      ddlItemType_SelectedIndexChanged
201  * Type:      Event Handler Method
202  * Purpose:   Handles showing and hiding the various controls to allow a user to enter
203  *            information so that an item can be added to the database.
204  * Input:     object sender, holds a reference to the object that raised this event.
205  * Input:     EventArgs e, holds data related to this event.
206  * Output:    Nothing.
207  -----*/
208  protected void ddlItemType_SelectedIndexChanged(object sender, EventArgs e)
209  {
210      // Hide things until needed.
211      lblError.Visible = false;
212      lblResult.Visible = false;
213
214      mViewBasic.ActiveViewIndex = -1;
215      mViewSpecific.ActiveViewIndex = -1;
216      btnAdd.Visible = false;
217
218      // Decide which controls should be shown to enable user to add an item.
219      switch ((Global.Enums.ItemTypes)ddlItemType.SelectedIndex)
220      {
221          case Global.Enums.ItemTypes.Business:
222              mViewBasic.ActiveViewIndex = 0;
223              mViewSpecific.ActiveViewIndex = 0;
224              lblName.Text = Global.Strings.BusinessName + Global.Strings.Separator;
225              lblType.Text = Global.Strings.BusinessType + Global.Strings.Separator;
226              btnAdd.Visible = true;
227              break;
228
229          case Global.Enums.ItemTypes.Park:
230              mViewBasic.ActiveViewIndex = 0;
231              mViewSpecific.ActiveViewIndex = 1;
232              lblName.Text = Global.Strings.ParkName + Global.Strings.Separator;
233              lblType.Text = Global.Strings.ParkType + Global.Strings.Separator;
234              btnAdd.Visible = true;
235              break;
236
237          case Global.Enums.ItemTypes.PublicFacility:
238              mViewBasic.ActiveViewIndex = 0;
239              mViewSpecific.ActiveViewIndex = -1;
240              lblName.Text = Global.Strings.PublicFacilityName + Global.Strings.Separator;
241              lblType.Text = Global.Strings.PublicFacilityType + Global.Strings.Separator;
242              btnAdd.Visible = true;
243              break;
244
245          default:
246              break;
247      }
248  }
249  }
250  }

```