

```
1 /*-----
2  * Author:      Dan Cassidy
3  * Date:        2015-06-23
4  * Assignment:  cView-P4
5  * Source File: Default.aspx.cs
6  * Language:    C#
7  * Course:      CSCI-C 490, C# Programming, Mowe 08:00
8  * Project:     The overall goal of this project is to capitalize on the fact that government, from
9  *              local to national, has made some of its data open by developing a way to explore
10 *              this data and present it to a user in a meaningful fashion. This phase of the
11 *              project is meant to explore data from any combination of the Business dataset
12 *              (https://data.southbendin.gov/d/imxu-7m5i), the Parks and Features dataset
13 *              (https://data.southbendin.gov/d/yf5x-7tkb), and the Public Facility dataset
14 *              (https://data.southbendin.gov/d/jeeff-dsq9) using a web-based ASP.NET UI interacting
15 *              with a SQL backend via the Entity Framework.
16 * Purpose:     Code-behind file for Default.aspx. This file's only purpose is to redirect the
17 *              client to the Main.aspx page.
18 -----*/
19
20 using System;
21 using System.Collections.Generic;
22 using System.Linq;
23 using System.Web;
24 using System.Web.UI;
25 using System.Web.UI.WebControls;
26
27 namespace cView_P4_DanCassidy
28 {
29     public partial class Default : System.Web.UI.Page
30     {
31         /*-----
32          * Name:      Page_Load
33          * Type:      Event Handler Method
34          * Purpose:   Handles anything that should happen on page load. In this case, it shunts the
35          *           client over to the Main.aspx page instead of this one.
36          * Input:    object sender, holds a reference to the object that raised this event.
37          * Input:    EventArgs e, holds data related to this event.
38          * Output:   Nothing.
39          -----*/
40         protected void Page_Load(object sender, EventArgs e)
41         {
42             Server.Transfer("~/Main.aspx");
43         }
44     }
45 }
```

```

1  /*-----
2  * Author:      Dan Cassidy
3  * Date:        2015-06-23
4  * Assignment:  cView-P4
5  * Source File: Main.aspx.cs
6  * Language:    C#
7  * Course:      CSCI-C 490, C# Programming, MoWe 08:00
8  * Purpose:     Code-behind file for Main.aspx. Controls randomization and resetting of the tables
9  *              behind the application.
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 Menu : System.Web.UI.Page
22     {
23         /*-----
24         * Name:      btnRandomize_Click
25         * Type:      Event Handler Method
26         * Purpose:   Handles randomizing the tables when clicked.
27         * Input:     object sender, holds a reference to the object that raised this event.
28         * Input:     EventArgs e, holds data related to this event.
29         * Output:    Nothing.
30         -----*/
31         protected void btnRandomize_Click(object sender, EventArgs e)
32         {
33             // Hide things until needed.
34             lblResult.Visible = false;
35             lblError.Visible = false;
36
37             try
38             {
39                 using (CViewDataEntities db = new CViewDataEntities())
40                 {
41                     // Used SQL statements because I didn't want to add more tables to the Entity
42                     // Framework model just for this. Also, it's easier.
43                     db.Database.ExecuteSqlCommand("TRUNCATE TABLE dbo.Business");
44                     db.Database.ExecuteSqlCommand("TRUNCATE TABLE dbo.BusinessReset");
45                     db.Database.ExecuteSqlCommand("TRUNCATE TABLE dbo.Park");
46                     db.Database.ExecuteSqlCommand("TRUNCATE TABLE dbo.ParkReset");
47                     db.Database.ExecuteSqlCommand("TRUNCATE TABLE dbo.PublicFacility");
48                     db.Database.ExecuteSqlCommand("TRUNCATE TABLE dbo.PublicFacilityReset");
49
50                     db.Database.ExecuteSqlCommand("INSERT INTO dbo.BusinessReset SELECT " +
51                         "TOP 50 * FROM dbo.BusinessBase ORDER BY NEWID()");
52                     db.Database.ExecuteSqlCommand("INSERT INTO dbo.Business SELECT " +
53                         " * FROM dbo.BusinessReset");
54                     db.Database.ExecuteSqlCommand("INSERT INTO dbo.ParkReset SELECT " +
55                         "TOP 50 * FROM dbo.ParkBase ORDER BY NEWID()");
56                     db.Database.ExecuteSqlCommand("INSERT INTO dbo.Park SELECT " +
57                         " * FROM dbo.ParkReset");
58                     db.Database.ExecuteSqlCommand("INSERT INTO dbo.PublicFacilityReset SELECT " +
59                         "TOP 50 * FROM dbo.PublicFacilityBase ORDER BY NEWID()");
60                     db.Database.ExecuteSqlCommand("INSERT INTO dbo.PublicFacility SELECT " +
61                         " * FROM dbo.PublicFacilityReset");
62                 }
63
64                 lblResult.Text = "The tables have been randomized.";
65                 lblResult.Visible = true;
66             }
67         }
68     }
69 }

```

```

67         catch
68         {
69             lblError.Text = "Error: Could not randomize the tables.";
70             lblError.Visible = true;
71         }
72     }
73
74     /*-----
75     * Name:      btnReset_Click
76     * Type:      Event Handler Method
77     * Purpose:   Handles resetting the tables to their prior randomized states.
78     * Input:     object sender, holds a reference to the object that raised this event.
79     * Input:     EventArgs e, holds data related to this event.
80     * Output:    Nothing.
81     -----*/
82     protected void btnReset_Click(object sender, EventArgs e)
83     {
84         // Hide things until needed.
85         lblResult.Visible = false;
86         lblError.Visible = false;
87
88         try
89         {
90             using (CViewDataEntities db = new CViewDataEntities())
91             {
92                 // Again, used SQL statements because I didn't want to add more tables to the
93                 // Entity Framework model just for this. Also, it's easier.
94                 db.Database.ExecuteSqlCommand("TRUNCATE TABLE dbo.Business");
95                 db.Database.ExecuteSqlCommand("TRUNCATE TABLE dbo.Park");
96                 db.Database.ExecuteSqlCommand("TRUNCATE TABLE dbo.PublicFacility");
97
98                 db.Database.ExecuteSqlCommand("INSERT INTO dbo.Business SELECT " +
99                     " * FROM dbo.BusinessReset");
100                db.Database.ExecuteSqlCommand("INSERT INTO dbo.Park SELECT " +
101                    " * FROM dbo.ParkReset");
102                db.Database.ExecuteSqlCommand("INSERT INTO dbo.PublicFacility SELECT " +
103                    " * FROM dbo.PublicFacilityReset");
104            }
105
106            lblResult.Text = "The tables have been reset to their prior randomized states.";
107            lblResult.Visible = true;
108        }
109        catch
110        {
111            lblError.Text = "Error: Could not reset the tables.";
112            lblError.Visible = true;
113        }
114    }
115 }
116 }

```

```

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 }

```

```

1  /*-----
2  * Author:      Dan Cassidy
3  * Date:        2015-06-23
4  * Assignment:  cView-P4
5  * Source File: Modify.aspx.cs
6  * Language:    C#
7  * Course:      CSCI-C 490, C# Programming, MoWe 08:00
8  * Purpose:     Code-behind file for Modify.aspx. Controls the process of modifying an item in the
9  *              database via the Entity Framework model.
10 -----*/
11
12 using System;
13 using System.Collections.Generic;
14 using System.Data.Entity;
15 using System.Linq;
16 using System.Web;
17 using System.Web.UI;
18 using System.Web.UI.WebControls;
19
20 namespace cView_P4_DanCassidy
21 {
22     public partial class Modify : System.Web.UI.Page
23     {
24         /*-----
25         * Name:      btnModify_Click
26         * Type:      Event Handler Method
27         * Purpose:   Handles showing and hiding the various controls to aid in allowing the user to
28         *            change the data associated with the chosen item.
29         * Input:     object sender, holds a reference to the object that raised this event.
30         * Input:     EventArgs e, holds data related to this event.
31         * Output:    Nothing.
32         -----*/
33         protected void btnModify_Click(object sender, EventArgs e)
34         {
35             // Quick check to make sure that something is selected.
36             switch ((Global.Enums.ItemTypes)ddlItemType.SelectedIndex)
37             {
38                 case Global.Enums.ItemTypes.Business:
39                     if (gViewBusiness.SelectedIndex == -1)
40                         return;
41                     break;
42
43                 case Global.Enums.ItemTypes.Park:
44                     if (gViewPark.SelectedIndex == -1)
45                         return;
46                     break;
47
48                 case Global.Enums.ItemTypes.PublicFacility:
49                     if (gViewPublicFacility.SelectedIndex == -1)
50                         return;
51                     break;
52
53                 default:
54                     break;
55             }
56
57             // Hide things until needed.
58             lblError.Visible = false;
59             lblResult.Visible = false;
60             mViewDisplay.ActiveViewIndex = -1;
61             mViewModifyBasic.ActiveViewIndex = -1;
62             mViewModifySpecific.ActiveViewIndex = -1;
63             btnModify.Visible = false;
64             btnSaveChanges.Visible = false;
65             btnBack.Visible = false;
66

```



```

67     try
68     {
69         using (CViewDataEntities database = new CViewDataEntities())
70         {
71             object keyToModify = null;
72
73             // Choose what to display based on the selected item type.
74             switch ((Global.Enums.ItemTypes)ddlItemType.SelectedIndex)
75             {
76                 case Global.Enums.ItemTypes.Business:
77                     // Basic UI prep.
78                     mViewModifyBasic.ActiveViewIndex = 0;
79                     mViewModifySpecific.ActiveViewIndex = 0;
80                     lblName.Text = Global.Strings.BusinessName + Global.Strings.Separator;
81                     lblType.Text = Global.Strings.BusinessType + Global.Strings.Separator;
82                     btnSaveChanges.Visible = true;
83                     btnBack.Visible = true;
84
85                     // Get object to modify.
86                     keyToModify = gViewBusiness.SelectedDataKey.Value;
87                     Business businessToModify = database.Businesses.Find(keyToModify);
88
89                     // Fill in textbox values.
90                     txtName.Text = businessToModify.Name;
91                     txtName.Enabled = true;
92                     txtType.Text = businessToModify.Type;
93                     txtStreetAddress.Text = businessToModify.StreetAddress;
94                     txtCity.Text = businessToModify.City;
95                     txtState.Text = businessToModify.State;
96                     txtZip.Text = businessToModify.Zip;
97                     txtLatitude.Text = businessToModify.Latitude.ToString();
98                     txtLongitude.Text = businessToModify.Longitude.ToString();
99                     txtPhone.Text = businessToModify.Phone;
100                     txtLicenseNumber.Text = businessToModify.LicenseNumber;
101                     txtLicenseIssueDate.Text = string.Format("{0:d}",
102                         businessToModify.LicenseIssueDate);
103                     txtLicenseExpirDate.Text = string.Format("{0:d}",
104                         businessToModify.LicenseExpirDate);
105                     txtLicenseStatus.Text = businessToModify.LicenseStatus;
106                     txtCouncilDistrict.Text = businessToModify.CouncilDistrict;
107                     break;
108
109                 case Global.Enums.ItemTypes.Park:
110                     // Basic UI prep.
111                     mViewModifyBasic.ActiveViewIndex = 0;
112                     mViewModifySpecific.ActiveViewIndex = 1;
113                     lblName.Text = Global.Strings.ParkName + Global.Strings.Separator;
114                     lblType.Text = Global.Strings.ParkType + Global.Strings.Separator;
115                     btnSaveChanges.Visible = true;
116                     btnBack.Visible = true;
117
118                     // Get object to modify.
119                     keyToModify = gViewPark.SelectedDataKey.Value;
120                     Park parkToModify = database.Parks.Find(keyToModify);
121
122                     // Fill in textbox values.
123                     txtName.Text = parkToModify.Name;
124                     txtName.Enabled = false;
125                     txtType.Text = parkToModify.Type;
126                     txtStreetAddress.Text = parkToModify.StreetAddress;
127                     txtCity.Text = parkToModify.City;
128                     txtState.Text = parkToModify.State;
129                     txtZip.Text = parkToModify.Zip;
130                     txtLatitude.Text = parkToModify.Latitude.ToString();
131                     txtLongitude.Text = parkToModify.Longitude.ToString();
132                     txtPhone.Text = parkToModify.Phone;

```

```

133         txtFeatureBaseball.Text = parkToModify.FeatureBaseball.ToString();
134         txtFeatureBasketball.Text = parkToModify.FeatureBasketball.ToString();
135         txtFeatureGolf.Text = parkToModify.FeatureGolf.ToString();
136         txtFeatureLargeMPField.Text = parkToModify.FeatureLargeMPField.
137             ToString();
138         txtFeatureTennis.Text = parkToModify.FeatureTennis.ToString();
139         txtFeatureVolleyball.Text = parkToModify.FeatureVolleyball.ToString();
140         break;
141
142     case Global.Enums.ItemTypes.PublicFacility:
143         // Basic UI prep.
144         mViewModifyBasic.ActiveViewIndex = 0;
145         lblName.Text = Global.Strings.PublicFacilityName +
146             Global.Strings.Separator;
147         lblType.Text = Global.Strings.PublicFacilityType +
148             Global.Strings.Separator;
149         btnSaveChanges.Visible = true;
150         btnBack.Visible = true;
151
152         // Get object to modify.
153         keyToModify = gViewPublicFacility.SelectedDataKey.Value;
154         PublicFacility publicFacilityToModify = database.PublicFacilities.
155             Find(keyToModify);
156
157         // Fill in textbox values.
158         txtName.Text = publicFacilityToModify.Name;
159         txtName.Enabled = false;
160         txtType.Text = publicFacilityToModify.Type;
161         txtStreetAddress.Text = publicFacilityToModify.StreetAddress;
162         txtCity.Text = publicFacilityToModify.City;
163         txtState.Text = publicFacilityToModify.State;
164         txtZip.Text = publicFacilityToModify.Zip;
165         txtLatitude.Text = publicFacilityToModify.Latitude.ToString();
166         txtLongitude.Text = publicFacilityToModify.Longitude.ToString();
167         txtPhone.Text = publicFacilityToModify.Phone;
168         break;
169
170     default:
171         throw new InvalidOperationException(
172             "Invalid item type dropdown value.");
173     }
174 }
175 }
176 catch (Exception ex)
177 {
178     lblError.Text = "Error: " + ex.Message;
179     lblError.Visible = true;
180 }
181 }
182
183 /*-----
184 * Name:     btnSaveChanges_Click
185 * Type:     Event Handler Method
186 * Purpose:  Handles saving the changed data for the specific item back to the database via
187 *           the Entity Framework model.
188 * Input:    object sender, holds a reference to the object that raised this event.
189 * Input:    EventArgs e, holds data related to this event.
190 * Output:   Nothing.
191 -----*/
192 protected void btnSaveChanges_Click(object sender, EventArgs e)
193 {
194     // Hide things until needed.
195     lblError.Visible = false;
196     lblResult.Visible = false;
197     mViewDisplay.ActiveViewIndex = -1;
198     mViewModifyBasic.ActiveViewIndex = -1;

```

```

199     mViewModifySpecific.ActiveViewIndex = -1;
200     btnModify.Visible = false;
201     btnSaveChanges.Visible = false;
202     btnBack.Visible = false;
203
204     try
205     {
206         using (CViewDataEntities database = new CViewDataEntities())
207         {
208             object keyToModify = null;
209
210             // Choose what to do based on the selected item type.
211             switch ((Global.Enums.ItemTypes)ddlItemType.SelectedIndex)
212             {
213                 case Global.Enums.ItemTypes.Business:
214                     // Get object to modify.
215                     keyToModify = gViewBusiness.SelectedDataKey.Value;
216                     Business businessToModify = database.Businesses.Find(keyToModify);
217
218                     // Save values to object.
219                     businessToModify.Name = txtName.Text.Trim();
220                     businessToModify.Type = txtType.Text.Trim();
221                     businessToModify.StreetAddress = txtStreetAddress.Text.Trim();
222                     businessToModify.City = txtCity.Text.Trim();
223                     businessToModify.State = txtState.Text.Trim();
224                     businessToModify.Zip = txtZip.Text.Trim();
225                     businessToModify.Latitude = SimpleConvert.ToDecimal(txtLatitude.Text.
226                         Trim());
227                     businessToModify.Longitude = SimpleConvert.ToDecimal(txtLongitude.Text.
228                         Trim());
229                     businessToModify.Phone = txtPhone.Text.Trim();
230                     businessToModify.LicenseIssueDate = SimpleConvert.ToDateTime(
231                         txtLicenseIssueDate.Text.Trim());
232                     businessToModify.LicenseExpirDate = SimpleConvert.ToDateTime(
233                         txtLicenseExpirDate.Text.Trim());
234                     businessToModify.LicenseStatus = txtLicenseStatus.Text.Trim();
235                     businessToModify.CouncilDistrict = txtCouncilDistrict.Text.Trim();
236
237                     // Modelled off http://stackoverflow.com/a/15339512.
238                     database.Businesses.Attach(businessToModify);
239                     database.Entry(businessToModify).State = EntityState.Modified;
240                     break;
241
242                 case Global.Enums.ItemTypes.Park:
243                     // Get object to modify.
244                     keyToModify = gViewPark.SelectedDataKey.Value;
245                     Park parkToModify = database.Parks.Find(keyToModify);
246
247                     // Save values to object.
248                     parkToModify.Type = txtType.Text.Trim();
249                     parkToModify.StreetAddress = txtStreetAddress.Text.Trim();
250                     parkToModify.City = txtCity.Text.Trim();
251                     parkToModify.State = txtState.Text.Trim();
252                     parkToModify.Zip = txtZip.Text.Trim();
253                     parkToModify.Latitude = SimpleConvert.ToDecimal(txtLatitude.Text.
254                         Trim());
255                     parkToModify.Longitude = SimpleConvert.ToDecimal(txtLongitude.Text.
256                         Trim());
257                     parkToModify.Phone = txtPhone.Text.Trim();
258                     parkToModify.FeatureBaseball = SimpleConvert.ToByte(
259                         txtFeatureBaseball.Text.Trim());
260                     parkToModify.FeatureBasketball = SimpleConvert.ToDecimal(
261                         txtFeatureBasketball.Text.Trim());
262                     parkToModify.FeatureGolf = SimpleConvert.ToDecimal(
263                         txtFeatureGolf.Text.Trim());
264                     parkToModify.FeatureLargeMPField = SimpleConvert.ToByte(

```

```

265         txtFeatureLargeMPField.Text.Trim());
266     parkToModify.FeatureTennis = SimpleConvert.ToByte(
267         txtFeatureTennis.Text.Trim());
268     parkToModify.FeatureVolleyball = SimpleConvert.ToByte(
269         txtFeatureVolleyball.Text.Trim());
270
271     // Modelled off http://stackoverflow.com/a/15339512.
272     database.Parks.Attach(parkToModify);
273     database.Entry(parkToModify).State = EntityState.Modified;
274     break;
275
276     case Global.Enums.ItemTypes.PublicFacility:
277         // Get object to modify.
278         keyToModify = gViewPublicFacility.SelectedDataKey.Value;
279         PublicFacility publicFacilityToModify = database.PublicFacilities.
280             Find(keyToModify);
281
282         // Save values to object.
283         publicFacilityToModify.Type = txtType.Text.Trim();
284         publicFacilityToModify.StreetAddress = txtStreetAddress.Text.Trim();
285         publicFacilityToModify.City = txtCity.Text.Trim();
286         publicFacilityToModify.State = txtState.Text.Trim();
287         publicFacilityToModify.Zip = txtZip.Text.Trim();
288         publicFacilityToModify.Latitude = SimpleConvert.ToDecimal(
289             txtLatitude.Text.Trim());
290         publicFacilityToModify.Longitude = SimpleConvert.ToDecimal(
291             txtLongitude.Text.Trim());
292         publicFacilityToModify.Phone = txtPhone.Text.Trim();
293
294         // Modelled off http://stackoverflow.com/a/15339512.
295         database.PublicFacilities.Attach(publicFacilityToModify);
296         database.Entry(publicFacilityToModify).State = EntityState.Modified;
297         break;
298
299     default:
300         throw new InvalidOperationException(
301             "Invalid item type dropdown value.");
302     }
303     // Save any changes to the database and refresh gridviews.
304     database.SaveChanges();
305     mViewDisplay.DataBind();
306 }
307 // Go back to item selection and let the user know the operation was successful.
308 ddlItemType_SelectedIndexChanged(sender, e);
309 lblResult.Text = "Item modified successfully.";
310 lblResult.Visible = true;
311 }
312 catch (Exception ex)
313 {
314     lblError.Text = "Error: " + ex.Message;
315     lblError.Visible = true;
316 }
317 }
318
319 /-----
320 * Name:     ddlItemType_SelectedIndexChanged
321 * Type:     Event Handler Method
322 * Purpose:  Handles showing and hiding the various controls to aid in allowing the user to
323 *           select an item to modify.
324 * Input:    object sender, holds a reference to the object that raised this event.
325 * Input:    EventArgs e, holds data related to this event.
326 * Output:   Nothing.
327 -----*/
328 protected void ddlItemType_SelectedIndexChanged(object sender, EventArgs e)
329 {
330     // Hide things until needed.

```

```
331         lblError.Visible = false;
332         lblResult.Visible = false;
333         mViewDisplay.ActiveViewIndex = -1;
334         mViewModifyBasic.ActiveViewIndex = -1;
335         mViewModifySpecific.ActiveViewIndex = -1;
336         btnModify.Visible = false;
337         btnSaveChanges.Visible = false;
338         btnBack.Visible = false;
339
340         // Reset selected indexes of GridView controls.
341         gridViewBusiness.SelectedIndex = -1;
342         gridViewPark.SelectedIndex = -1;
343         gridViewPublicFacility.SelectedIndex = -1;
344
345         // Show the appropriate view.
346         switch ((Global.Enums.ItemTypes)ddlItemType.SelectedIndex)
347         {
348             case Global.Enums.ItemTypes.Business:
349                 mViewDisplay.ActiveViewIndex = 0;
350                 btnModify.Visible = true;
351                 break;
352
353             case Global.Enums.ItemTypes.Park:
354                 mViewDisplay.ActiveViewIndex = 1;
355                 btnModify.Visible = true;
356                 break;
357
358             case Global.Enums.ItemTypes.PublicFacility:
359                 mViewDisplay.ActiveViewIndex = 2;
360                 btnModify.Visible = true;
361                 break;
362
363             default:
364                 break;
365         }
366     }
367 }
368 }
```

```

1  /*-----
2  * Author:      Dan Cassidy
3  * Date:        2015-06-23
4  * Assignment:  cView-P4
5  * Source File: Delete.aspx.cs
6  * Language:    C#
7  * Course:      CSCI-C 490, C# Programming, MoWe 08:00
8  * Purpose:     Code-behind file for Delete.aspx. Controls the deletion of a single row of data (1
9  *              object) from a table.
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 Delete : System.Web.UI.Page
22     {
23         /*-----
24         * Name:      btnDelete_Click
25         * Type:      Event Handler Method
26         * Purpose:   Handles deleting an object from the database.
27         * Input:     object sender, holds a reference to the object that raised this event.
28         * Input:     EventArgs e, holds data related to this event.
29         * Output:    Nothing.
30         -----*/
31         protected void btnDelete_Click(object sender, EventArgs e)
32         {
33             // Dummy polymorphoc object.
34             object keyToDelete = null;
35
36             // Attempt to delete a record.
37             try
38             {
39                 using (CViewDataEntities database = new CViewDataEntities())
40                 {
41                     switch ((Global.Enums.ItemTypes)ddlItemType.SelectedIndex)
42                     {
43                         case Global.Enums.ItemTypes.Business:
44                             if (gViewBusiness.SelectedIndex == -1)
45                                 return;
46                             keyToDelete = gViewBusiness.SelectedDataKey.Value;
47                             gViewBusiness.SelectRow(-1);
48                             Business businessToDelete = database.Businesses.Find(keyToDelete);
49                             database.Businesses.Remove(businessToDelete);
50                             break;
51
52                         case Global.Enums.ItemTypes.Park:
53                             if (gViewPark.SelectedIndex == -1)
54                                 return;
55                             keyToDelete = gViewPark.SelectedDataKey.Value;
56                             gViewPark.SelectRow(-1);
57                             Park parkToDelete = database.Parks.Find(keyToDelete);
58                             database.Parks.Remove(parkToDelete);
59                             break;
60
61                         case Global.Enums.ItemTypes.PublicFacility:
62                             if (gViewPublicFacility.SelectedIndex == -1)
63                                 return;
64                             keyToDelete = gViewPublicFacility.SelectedDataKey.Value;
65                             gViewPublicFacility.SelectRow(-1);
66                             PublicFacility publicFacilityToDelete =

```

```

67         database.PublicFacilities.Find(keyToDelete);
68         database.PublicFacilities.Remove(publicFacilityToDelete);
69         break;
70
71         default:
72             // ... How?
73             return;
74     }
75
76     database.SaveChanges();
77     mViewData.DataBind();
78 }
79
80 lblResult.Text = "Row successfully deleted.";
81 lblResult.Visible = true;
82 lblError.Visible = false;
83 }
84 catch
85 {
86     lblError.Text = "Error: Could not delete the specified row.";
87     lblError.Visible = true;
88     lblResult.Visible = false;
89 }
90 }
91
92 /*-----
93 * Name:     ddlItemType_SelectedIndexChanged
94 * Type:     Event Handler Method
95 * Purpose:  Handles showing and hiding the various controls to allow a user to choose an
96 *           item to be deleted from the database.
97 * Input:    object sender, holds a reference to the object that raised this event.
98 * Input:    EventArgs e, holds data related to this event.
99 * Output:   Nothing.
100 -----*/
101 protected void ddlItemType_SelectedIndexChanged(object sender, EventArgs e)
102 {
103     // Hide things until needed.
104     lblError.Visible = false;
105     lblResult.Visible = false;
106
107     mViewData.ActiveViewIndex = -1;
108     btnDelete.Visible = false;
109
110     // Decide which controls should be shown to enable user to delete an item.
111     switch ((Global.Enums.ItemTypes)ddlItemType.SelectedIndex)
112     {
113         case Global.Enums.ItemTypes.Business:
114             mViewData.ActiveViewIndex = 0;
115             btnDelete.Visible = true;
116             break;
117
118         case Global.Enums.ItemTypes.Park:
119             mViewData.ActiveViewIndex = 1;
120             btnDelete.Visible = true;
121             break;
122
123         case Global.Enums.ItemTypes.PublicFacility:
124             mViewData.ActiveViewIndex = 2;
125             btnDelete.Visible = true;
126             break;
127
128         default:
129             break;
130     }
131 }
132

```

```
133     }  
134 }
```



```

1  /*-----
2  * Author:      Dan Cassidy
3  * Date:        2015-06-23
4  * Assignment:  cView-P4
5  * Source File: Display.aspx.cs
6  * Language:    C#
7  * Course:      CSCI-C 490, C# Programming, Mowe 08:00
8  * Purpose:     Code-behind file for Display.aspx. Controls the display of the different tables of
9  *              items.
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 Display : System.Web.UI.Page
22     {
23         /*-----
24         * Name:      ddlItemType_SelectedIndexChanged
25         * Type:      Event Handler Method
26         * Purpose:   Handles showing and hiding the data views.
27         * Input:     object sender, holds a reference to the object that raised this event.
28         * Input:     EventArgs e, holds data related to this event.
29         * Output:    Nothing.
30         -----*/
31         protected void ddlItemType_SelectedIndexChanged(object sender, EventArgs e)
32         {
33             switch ((Global.Enums.ItemTypes)ddlItemType.SelectedIndex)
34             {
35                 case Global.Enums.ItemTypes.Business:
36                     mViewData.ActiveViewIndex = 0;
37                     break;
38
39                 case Global.Enums.ItemTypes.Park:
40                     mViewData.ActiveViewIndex = 1;
41                     break;
42
43                 case Global.Enums.ItemTypes.PublicFacility:
44                     mViewData.ActiveViewIndex = 2;
45                     break;
46
47                 default:
48                     mViewData.ActiveViewIndex = -1;
49                     break;
50             }
51         }
52     }
53 }

```

```

1  /*-----
2  * Author:      Dan Cassidy
3  * Date:        2015-06-23
4  * Assignment:  cView-P4
5  * Source File: Search.aspx.cs
6  * Language:    C#
7  * Course:      CSCI-C 490, C# Programming, MoWe 08:00
8  * Purpose:     Code-behind file for Search.aspx. Controls the interface whereby a user can search
9  *              the tables.
10 -----*/
11
12 using System;
13 using System.Collections.Generic;
14 using System.Data.Entity;
15 using System.Linq;
16 using System.Web;
17 using System.Web.UI;
18 using System.Web.UI.WebControls;
19
20 namespace cView_P4_DanCassidy
21 {
22     public partial class Search : System.Web.UI.Page
23     {
24         /*-----
25         * Name:      btnSearch_Click
26         * Type:      Event Handler Method
27         * Purpose:   Handles the actual search and display of the results.
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 btnSearch_Click(object sender, EventArgs e)
33         {
34             // Hide things until needed.
35             lblError.Visible = false;
36             lblResult.Visible = false;
37             mViewSearchResults.ActiveViewIndex = -1;
38
39             // Pre-conversions to save some space.
40             string toSearchFor = txtSearch.Text.Trim();
41             byte toSearchForByte = SimpleConvert.ToByte(toSearchFor);
42             DateTime toSearchForDateTime = SimpleConvert.ToDateTime(toSearchFor);
43             decimal toSearchForDecimal = SimpleConvert.ToDecimal(toSearchFor);
44
45             int resultCount = 0;
46             int viewToDisplay = -1;
47             GridView gViewToDisplay;
48
49             int selectedIndex;
50             object baseObject;
51
52             // Determine what comparator to use; default is "|" for "contains".
53             string comparator = "|";
54             if (ddlComparatorsStrings.Visible == true)
55             {
56                 switch ((Global.Enums.ComparatorsStrings)ddlComparatorsStrings.SelectedIndex)
57                 {
58                     case Global.Enums.ComparatorsStrings.NotContain:
59                         comparator = "!=";
60                         break;
61
62                     case Global.Enums.ComparatorsStrings.Equal:
63                         comparator = "==";
64                         break;
65
66                     case Global.Enums.ComparatorsStrings.NotEqual:

```

```

67         comparator = "!=";
68         break;
69
70     default:
71         break;
72     }
73 }
74 else if (ddlComparatorsNotStrings.Visible == true)
75 {
76     switch ((Global.Enums.ComparatorsNotStrings)ddlComparatorsNotStrings.SelectedIndex)
77     {
78         case Global.Enums.ComparatorsNotStrings.NotContain:
79             comparator = "!=";
80             break;
81
82         case Global.Enums.ComparatorsNotStrings.Equal:
83             comparator = "==";
84             break;
85
86         case Global.Enums.ComparatorsNotStrings.NotEqual:
87             comparator = "!=";
88             break;
89
90         case Global.Enums.ComparatorsNotStrings.Greater:
91             comparator = ">";
92             break;
93
94         case Global.Enums.ComparatorsNotStrings.Less:
95             comparator = "<";
96             break;
97
98         case Global.Enums.ComparatorsNotStrings.GreaterEqual:
99             comparator = ">=";
100            break;
101
102         case Global.Enums.ComparatorsNotStrings.LessEqual:
103             comparator = "<=";
104             break;
105
106         default:
107             break;
108     }
109 }
110
111 // Do the search.
112 //
113 // The method I ended up using is a bit kludgy, but I had to resort to this because LINQ
114 // to Entities is stupid and doesn't even try to evaluate what it can server-side prior
115 // to attempting to translate the query to SQL and failing because indexers are not a
116 // SQL thing.
117 try
118 {
119     using (CViewDataEntities database = new CViewDataEntities())
120     {
121         switch ((Global.Enums.ItemTypes)ddlItemType.SelectedIndex)
122         {
123             case Global.Enums.ItemTypes.Business:
124                 viewToDisplay = 0;
125                 gViewToDisplay = gViewBusinessResults;
126                 selectedIndex = ddlBusiness.SelectedIndex;
127                 baseObject = database.Businesses.First()[selectedIndex];
128                 if (baseObject != null)
129                 {
130                     IEnumerable<Business> searchResults = null;
131                     DbSet<Business> tableToSearch = database.Businesses;
132                     switch (comparator)

```

```

133     {
134     case "|":
135         searchResults = tableToSearch.AsEnumerable().Where(i =>
136             i[selectedIndex].ToString().IndexOf(
137                 toSearchFor, StringComparison.OrdinalIgnoreCase) >= 0);
138         break;
139
140     case "!!":
141         searchResults = tableToSearch.AsEnumerable().Where(i =>
142             i[selectedIndex].ToString().IndexOf(
143                 toSearchFor, StringComparison.OrdinalIgnoreCase) < 0);
144         break;
145
146     case "==" :
147         if (baseObject is byte)
148             searchResults = tableToSearch.AsEnumerable().Where(i =>
149                 (byte)i[selectedIndex] ==
150                 toSearchForByte);
151         else if (baseObject is DateTime)
152             searchResults = tableToSearch.AsEnumerable().Where(i =>
153                 (DateTime)i[selectedIndex] ==
154                 toSearchForDateTime);
155         else if (baseObject is decimal)
156             searchResults = tableToSearch.AsEnumerable().Where(i =>
157                 (decimal)i[selectedIndex] ==
158                 toSearchForDecimal);
159         else
160             searchResults = tableToSearch.AsEnumerable().Where(i =>
161                 ((string)i[selectedIndex]).ToLower() == toSearchFor.
162                 ToLower());
163         break;
164
165     case "!=" :
166         if (baseObject is byte)
167             searchResults = tableToSearch.AsEnumerable().Where(i =>
168                 (byte)i[selectedIndex] !=
169                 toSearchForByte);
170         else if (baseObject is DateTime)
171             searchResults = tableToSearch.AsEnumerable().Where(i =>
172                 (DateTime)i[selectedIndex] !=
173                 toSearchForDateTime);
174         else if (baseObject is decimal)
175             searchResults = tableToSearch.AsEnumerable().Where(i =>
176                 (decimal)i[selectedIndex] !=
177                 toSearchForDecimal);
178         else
179             searchResults = tableToSearch.AsEnumerable().Where(i =>
180                 ((string)i[selectedIndex]).ToLower() != toSearchFor.
181                 ToLower());
182         break;
183
184     case ">" :
185         if (baseObject is byte)
186             searchResults = tableToSearch.AsEnumerable().Where(i =>
187                 (byte)i[selectedIndex] >
188                 toSearchForByte);
189         else if (baseObject is DateTime)
190             searchResults = tableToSearch.AsEnumerable().Where(i =>
191                 (DateTime)i[selectedIndex] >
192                 toSearchForDateTime);
193         else if (baseObject is decimal)
194             searchResults = tableToSearch.AsEnumerable().Where(i =>
195                 (decimal)i[selectedIndex] >
196                 toSearchForDecimal);
197         else
198             throw new InvalidOperationException(

```

```

199         "Comparator \">>\" cannot be applied to strings.");
200     break;
201
202     case "<":
203         if (baseObject is byte)
204             searchResults = tableToSearch.AsEnumerable().Where(i =>
205                 (byte)i[selectedIndex] <
206                 toSearchForByte);
207         else if (baseObject is DateTime)
208             searchResults = tableToSearch.AsEnumerable().Where(i =>
209                 (DateTime)i[selectedIndex] <
210                 toSearchForDateTime);
211         else if (baseObject is decimal)
212             searchResults = tableToSearch.AsEnumerable().Where(i =>
213                 (decimal)i[selectedIndex] <
214                 toSearchForDecimal);
215         else
216             throw new InvalidOperationException(
217                 "Comparator \"><\" cannot be applied to strings.");
218     break;
219
220     case ">=":
221         if (baseObject is byte)
222             searchResults = tableToSearch.AsEnumerable().Where(i =>
223                 (byte)i[selectedIndex] >=
224                 toSearchForByte);
225         else if (baseObject is DateTime)
226             searchResults = tableToSearch.AsEnumerable().Where(i =>
227                 (DateTime)i[selectedIndex] >=
228                 toSearchForDateTime);
229         else if (baseObject is decimal)
230             searchResults = tableToSearch.AsEnumerable().Where(i =>
231                 (decimal)i[selectedIndex] >=
232                 toSearchForDecimal);
233         else
234             throw new InvalidOperationException(
235                 "Comparator \">>=\\" cannot be applied to strings.");
236     break;
237
238     case "<=":
239         if (baseObject is byte)
240             searchResults = tableToSearch.AsEnumerable().Where(i =>
241                 (byte)i[selectedIndex] <=
242                 toSearchForByte);
243         else if (baseObject is DateTime)
244             searchResults = tableToSearch.AsEnumerable().Where(i =>
245                 (DateTime)i[selectedIndex] <=
246                 toSearchForDateTime);
247         else if (baseObject is decimal)
248             searchResults = tableToSearch.AsEnumerable().Where(i =>
249                 (decimal)i[selectedIndex] <=
250                 toSearchForDecimal);
251         else
252             throw new InvalidOperationException(
253                 "Comparator \"><=\\" cannot be applied to strings.");
254     break;
255
256     default:
257         throw new InvalidOperationException("Invalid comparator.");
258 }
259 resultCount = searchResults.Count();
260 gViewToDisplay.DataSource = searchResults.ToList();
261 }
262 break;
263
264 case Global.Enums.ItemTypes.Park:

```

```

265         viewToDisplay = 1;
266         gViewToDisplay = gViewParkResults;
267         selectedIndex = ddlPark.SelectedIndex;
268         baseObject = database.Parks.First()[selectedIndex];
269         if (baseObject != null)
270         {
271             IEnumerable<Park> searchResults = null;
272             DbSet<Park> tableToSearch = database.Parks;
273             switch (comparator)
274             {
275                 case "|":
276                     searchResults = tableToSearch.AsEnumerable().Where(i =>
277                         i[selectedIndex].ToString().IndexOf(
278                             toSearchFor, StringComparison.OrdinalIgnoreCase) >= 0);
279                     break;
280
281                 case "||":
282                     searchResults = tableToSearch.AsEnumerable().Where(i =>
283                         i[selectedIndex].ToString().IndexOf(
284                             toSearchFor, StringComparison.OrdinalIgnoreCase) < 0);
285                     break;
286
287                 case "==":
288                     if (baseObject is byte)
289                         searchResults = tableToSearch.AsEnumerable().Where(i =>
290                             (byte)i[selectedIndex] ==
291                             toSearchForByte);
292                     else if (baseObject is DateTime)
293                         searchResults = tableToSearch.AsEnumerable().Where(i =>
294                             (DateTime)i[selectedIndex] ==
295                             toSearchForDateTime);
296                     else if (baseObject is decimal)
297                         searchResults = tableToSearch.AsEnumerable().Where(i =>
298                             (decimal)i[selectedIndex] ==
299                             toSearchForDecimal);
300                     else
301                         searchResults = tableToSearch.AsEnumerable().Where(i =>
302                             ((string)i[selectedIndex]).ToLower() == toSearchFor.
303                             ToLower());
304                     break;
305
306                 case "!=":
307                     if (baseObject is byte)
308                         searchResults = tableToSearch.AsEnumerable().Where(i =>
309                             (byte)i[selectedIndex] !=
310                             toSearchForByte);
311                     else if (baseObject is DateTime)
312                         searchResults = tableToSearch.AsEnumerable().Where(i =>
313                             (DateTime)i[selectedIndex] !=
314                             toSearchForDateTime);
315                     else if (baseObject is decimal)
316                         searchResults = tableToSearch.AsEnumerable().Where(i =>
317                             (decimal)i[selectedIndex] !=
318                             toSearchForDecimal);
319                     else
320                         searchResults = tableToSearch.AsEnumerable().Where(i =>
321                             ((string)i[selectedIndex]).ToLower() != toSearchFor.
322                             ToLower());
323                     break;
324
325                 case ">":
326                     if (baseObject is byte)
327                         searchResults = tableToSearch.AsEnumerable().Where(i =>
328                             (byte)i[selectedIndex] >
329                             toSearchForByte);
330                     else if (baseObject is DateTime)

```

```

331         searchResults = tableToSearch.AsEnumerable().Where(i =>
332             (DateTime)i[selectedIndex] >
333             toSearchForDateTime);
334     else if (baseObject is decimal)
335         searchResults = tableToSearch.AsEnumerable().Where(i =>
336             (decimal)i[selectedIndex] >
337             toSearchForDecimal);
338     else
339         throw new InvalidOperationException(
340             "Comparator \">>\" cannot be applied to strings.");
341     break;
342
343     case "<":
344         if (baseObject is byte)
345             searchResults = tableToSearch.AsEnumerable().Where(i =>
346                 (byte)i[selectedIndex] <
347                 toSearchForByte);
348         else if (baseObject is DateTime)
349             searchResults = tableToSearch.AsEnumerable().Where(i =>
350                 (DateTime)i[selectedIndex] <
351                 toSearchForDateTime);
352         else if (baseObject is decimal)
353             searchResults = tableToSearch.AsEnumerable().Where(i =>
354                 (decimal)i[selectedIndex] <
355                 toSearchForDecimal);
356         else
357             throw new InvalidOperationException(
358                 "Comparator \"><\" cannot be applied to strings.");
359         break;
360
361     case ">=":
362         if (baseObject is byte)
363             searchResults = tableToSearch.AsEnumerable().Where(i =>
364                 (byte)i[selectedIndex] >=
365                 toSearchForByte);
366         else if (baseObject is DateTime)
367             searchResults = tableToSearch.AsEnumerable().Where(i =>
368                 (DateTime)i[selectedIndex] >=
369                 toSearchForDateTime);
370         else if (baseObject is decimal)
371             searchResults = tableToSearch.AsEnumerable().Where(i =>
372                 (decimal)i[selectedIndex] >=
373                 toSearchForDecimal);
374         else
375             throw new InvalidOperationException(
376                 "Comparator \">>=\" cannot be applied to strings.");
377         break;
378
379     case "<=":
380         if (baseObject is byte)
381             searchResults = tableToSearch.AsEnumerable().Where(i =>
382                 (byte)i[selectedIndex] <=
383                 toSearchForByte);
384         else if (baseObject is DateTime)
385             searchResults = tableToSearch.AsEnumerable().Where(i =>
386                 (DateTime)i[selectedIndex] <=
387                 toSearchForDateTime);
388         else if (baseObject is decimal)
389             searchResults = tableToSearch.AsEnumerable().Where(i =>
390                 (decimal)i[selectedIndex] <=
391                 toSearchForDecimal);
392         else
393             throw new InvalidOperationException(
394                 "Comparator \"><=\" cannot be applied to strings.");
395         break;
396

```

```

397         default:
398             throw new InvalidOperationException("Invalid comparator.");
399     }
400     resultCount = searchResults.Count();
401     gridViewToDisplay.DataSource = searchResults.ToList();
402 }
403 break;
404
405 case Global.Enums.ItemTypes.PublicFacility:
406     viewToDisplay = 2;
407     gridViewToDisplay = gridViewPublicFacilityResults;
408     selectedIndex = ddlPublicFacility.SelectedIndex;
409     baseObject = database.PublicFacilities.First()[selectedIndex];
410     if (baseObject != null)
411     {
412         IEnumerable<PublicFacility> searchResults = null;
413         DbSet<PublicFacility> tableToSearch = database.PublicFacilities;
414         switch (comparator)
415         {
416             case "|":
417                 searchResults = tableToSearch.AsEnumerable().Where(i =>
418                     i[selectedIndex].ToString().IndexOf(
419                         toSearchFor, StringComparison.OrdinalIgnoreCase) >= 0);
420                 break;
421
422             case "!!":
423                 searchResults = tableToSearch.AsEnumerable().Where(i =>
424                     i[selectedIndex].ToString().IndexOf(
425                         toSearchFor, StringComparison.OrdinalIgnoreCase) < 0);
426                 break;
427
428             case "==" :
429                 if (baseObject is byte)
430                     searchResults = tableToSearch.AsEnumerable().Where(i =>
431                         (byte)i[selectedIndex] ==
432                         toSearchForByte);
433                 else if (baseObject is DateTime)
434                     searchResults = tableToSearch.AsEnumerable().Where(i =>
435                         (DateTime)i[selectedIndex] ==
436                         toSearchForDateTime);
437                 else if (baseObject is decimal)
438                     searchResults = tableToSearch.AsEnumerable().Where(i =>
439                         (decimal)i[selectedIndex] ==
440                         toSearchForDecimal);
441                 else
442                     searchResults = tableToSearch.AsEnumerable().Where(i =>
443                         ((string)i[selectedIndex]).ToLower() == toSearchFor.
444                         ToLower());
445                 break;
446
447             case "!=" :
448                 if (baseObject is byte)
449                     searchResults = tableToSearch.AsEnumerable().Where(i =>
450                         (byte)i[selectedIndex] !=
451                         toSearchForByte);
452                 else if (baseObject is DateTime)
453                     searchResults = tableToSearch.AsEnumerable().Where(i =>
454                         (DateTime)i[selectedIndex] !=
455                         toSearchForDateTime);
456                 else if (baseObject is decimal)
457                     searchResults = tableToSearch.AsEnumerable().Where(i =>
458                         (decimal)i[selectedIndex] !=
459                         toSearchForDecimal);
460                 else
461                     searchResults = tableToSearch.AsEnumerable().Where(i =>
462                         ((string)i[selectedIndex]).ToLower() != toSearchFor.

```



```

463         ToLower());
464     break;
465
466     case ">":
467         if (baseObject is byte)
468             searchResults = tableToSearch.AsEnumerable().Where(i =>
469                 (byte)i[selectedIndex] >
470                 toSearchForByte);
471         else if (baseObject is DateTime)
472             searchResults = tableToSearch.AsEnumerable().Where(i =>
473                 (DateTime)i[selectedIndex] >
474                 toSearchForDateTime);
475         else if (baseObject is decimal)
476             searchResults = tableToSearch.AsEnumerable().Where(i =>
477                 (decimal)i[selectedIndex] >
478                 toSearchForDecimal);
479         else
480             throw new InvalidOperationException(
481                 "Comparator \">>\" cannot be applied to strings.");
482     break;
483
484     case "<":
485         if (baseObject is byte)
486             searchResults = tableToSearch.AsEnumerable().Where(i =>
487                 (byte)i[selectedIndex] <
488                 toSearchForByte);
489         else if (baseObject is DateTime)
490             searchResults = tableToSearch.AsEnumerable().Where(i =>
491                 (DateTime)i[selectedIndex] <
492                 toSearchForDateTime);
493         else if (baseObject is decimal)
494             searchResults = tableToSearch.AsEnumerable().Where(i =>
495                 (decimal)i[selectedIndex] <
496                 toSearchForDecimal);
497         else
498             throw new InvalidOperationException(
499                 "Comparator \"><\" cannot be applied to strings.");
500     break;
501
502     case ">=":
503         if (baseObject is byte)
504             searchResults = tableToSearch.AsEnumerable().Where(i =>
505                 (byte)i[selectedIndex] >=
506                 toSearchForByte);
507         else if (baseObject is DateTime)
508             searchResults = tableToSearch.AsEnumerable().Where(i =>
509                 (DateTime)i[selectedIndex] >=
510                 toSearchForDateTime);
511         else if (baseObject is decimal)
512             searchResults = tableToSearch.AsEnumerable().Where(i =>
513                 (decimal)i[selectedIndex] >=
514                 toSearchForDecimal);
515         else
516             throw new InvalidOperationException(
517                 "Comparator \">>=\" cannot be applied to strings.");
518     break;
519
520     case "<=":
521         if (baseObject is byte)
522             searchResults = tableToSearch.AsEnumerable().Where(i =>
523                 (byte)i[selectedIndex] <=
524                 toSearchForByte);
525         else if (baseObject is DateTime)
526             searchResults = tableToSearch.AsEnumerable().Where(i =>
527                 (DateTime)i[selectedIndex] <=
528                 toSearchForDateTime);

```

```

529         else if (baseObject is decimal)
530             searchResults = tableToSearch.AsEnumerable().Where(i =>
531                 (decimal)i[selectedIndex] <=
532                 toSearchForDecimal);
533         else
534             throw new InvalidOperationException(
535                 "Comparator \"<=\" cannot be applied to strings.");
536         break;
537
538         default:
539             throw new InvalidOperationException("Invalid comparator.");
540     }
541     resultCount = searchResults.Count();
542     gridViewToDisplay.DataSource = searchResults.ToList();
543 }
544 break;
545
546 default:
547     throw new InvalidOperationException(
548         "Invalid item type dropdown value.");
549 }
550 lblResult.Text = string.Format("{0} result{1} found.", resultCount,
551     resultCount == 1 ? "" : "s");
552 lblResult.Visible = true;
553 mViewSearchResults.ActiveViewIndex = viewToDisplay;
554 gridViewToDisplay.DataBind();
555 }
556 }
557 catch (Exception ex)
558 {
559     lblError.Text = "Error: " + ex.Message;
560     lblError.Visible = true;
561 }
562 }
563
564 /*-----
565 * Name:     ddlBusiness_SelectedIndexChanged
566 * Type:     Event Handler Method
567 * Purpose:  Handles the change of visibility on controls when the user chooses a business
568 *           field.
569 * Input:    object sender, holds a reference to the object that raised this event.
570 * Input:    EventArgs e, holds data related to this event.
571 * Output:   Nothing.
572 -----*/
573 protected void ddlBusiness_SelectedIndexChanged(object sender, EventArgs e)
574 {
575     // Hide things until needed.
576     lblError.Visible = false;
577     lblResult.Visible = false;
578
579     ddlComparatorsStrings.Visible = false;
580     ddlComparatorsStrings.SelectedIndex = 0;
581     ddlComparatorsNotStrings.Visible = false;
582     ddlComparatorsNotStrings.SelectedIndex = 0;
583
584     txtSearch.Visible = false;
585     txtSearch.Text = "";
586     btnSearch.Visible = false;
587
588     mViewSearchResults.ActiveViewIndex = -1;
589
590     // Display the needed control.
591     switch ((Global.Enums.BusinessFields)ddlBusiness.SelectedIndex)
592     {
593     case Global.Enums.BusinessFields.Name:
594     case Global.Enums.BusinessFields.Type:

```

```

595         case Global.Enums.BusinessFields.StreetAddress:
596         case Global.Enums.BusinessFields.City:
597         case Global.Enums.BusinessFields.State:
598         case Global.Enums.BusinessFields.Zip:
599         case Global.Enums.BusinessFields.Phone:
600         case Global.Enums.BusinessFields.LicenseNumber:
601         case Global.Enums.BusinessFields.LicenseStatus:
602         case Global.Enums.BusinessFields.CouncilDistrict:
603             ddlComparatorsStrings.Visible = true;
604             break;
605
606         case Global.Enums.BusinessFields.Latitude:
607         case Global.Enums.BusinessFields.Longitude:
608         case Global.Enums.BusinessFields.LicenseIssueDate:
609         case Global.Enums.BusinessFields.LicenseExpirDate:
610             ddlComparatorsNotStrings.Visible = true;
611             break;
612
613         default:
614             ddlComparatorsStrings.Visible = false;
615             ddlComparatorsNotStrings.Visible = false;
616             break;
617     }
618 }
619
620 /*-----
621  * Name:      ddlComparatorsNotStrings_SelectedIndexChanged
622  * Type:      Event Handler Method
623  * Purpose:   Handles the change of visibility on controls when the user chooses what type of
624  *            comparison to make.
625  * Input:     object sender, holds a reference to the object that raised this event.
626  * Input:     EventArgs e, holds data related to this event.
627  * Output:    Nothing.
628  *-----*/
629 protected void ddlComparatorsNotStrings_SelectedIndexChanged(object sender, EventArgs e)
630 {
631     // Hide things until needed.
632     lblError.Visible = false;
633     lblResult.Visible = false;
634
635     txtSearch.Visible = false;
636     txtSearch.Text = "";
637     btnSearch.Visible = false;
638
639     mViewSearchResults.ActiveViewIndex = -1;
640
641     // Display the needed control.
642     switch ((Global.Enums.ComparatorsNotStrings)ddlComparatorsNotStrings.SelectedIndex)
643     {
644         case Global.Enums.ComparatorsNotStrings.Contain:
645         case Global.Enums.ComparatorsNotStrings.NotContain:
646         case Global.Enums.ComparatorsNotStrings.Equal:
647         case Global.Enums.ComparatorsNotStrings.NotEqual:
648         case Global.Enums.ComparatorsNotStrings.Greater:
649         case Global.Enums.ComparatorsNotStrings.Less:
650         case Global.Enums.ComparatorsNotStrings.GreaterEqual:
651         case Global.Enums.ComparatorsNotStrings.LessEqual:
652             txtSearch.Visible = true;
653             btnSearch.Visible = true;
654             break;
655
656         default:
657             break;
658     }
659 }
660

```

```

661  /*-----
662  * Name:      ddlComparatorsStrings_SelectedIndexChanged
663  * Type:      Event Handler Method
664  * Purpose:   Handles the change of visibility on controls when the user chooses what type of
665  *            comparison to make.
666  * Input:     object sender, holds a reference to the object that raised this event.
667  * Input:     EventArgs e, holds data related to this event.
668  * Output:    Nothing.
669  -----*/
670  protected void ddlComparatorsStrings_SelectedIndexChanged(object sender, EventArgs e)
671  {
672      // Hide things until needed.
673      lblError.Visible = false;
674      lblResult.Visible = false;
675
676      txtSearch.Visible = false;
677      txtSearch.Text = "";
678      btnSearch.Visible = false;
679
680      mViewSearchResults.ActiveViewIndex = -1;
681
682      // Display the needed control.
683      switch ((Global.Enums.ComparatorsStrings)ddlComparatorsStrings.SelectedIndex)
684      {
685          case Global.Enums.ComparatorsStrings.Contain:
686          case Global.Enums.ComparatorsStrings.NotContain:
687          case Global.Enums.ComparatorsStrings.Equal:
688          case Global.Enums.ComparatorsStrings.NotEqual:
689              txtSearch.Visible = true;
690              btnSearch.Visible = true;
691              break;
692
693          default:
694              break;
695      }
696  }
697
698  /*-----
699  * Name:      ddlItemType_SelectedIndexChanged
700  * Type:      Event Handler Method
701  * Purpose:   Handles showing and hiding the various controls to allow a user to search the
702  *            database.
703  * Input:     object sender, holds a reference to the object that raised this event.
704  * Input:     EventArgs e, holds data related to this event.
705  * Output:    Nothing.
706  -----*/
707  protected void ddlItemType_SelectedIndexChanged(object sender, EventArgs e)
708  {
709      // Hide things until needed.
710      lblError.Visible = false;
711      lblResult.Visible = false;
712
713      ddlBusiness.Visible = false;
714      ddlBusiness.SelectedIndex = 0;
715      ddlPark.Visible = false;
716      ddlPark.SelectedIndex = 0;
717      ddlPublicFacility.Visible = false;
718      ddlPublicFacility.SelectedIndex = 0;
719      ddlComparatorsStrings.Visible = false;
720      ddlComparatorsStrings.SelectedIndex = 0;
721      ddlComparatorsNotStrings.Visible = false;
722      ddlComparatorsNotStrings.SelectedIndex = 0;
723
724      txtSearch.Visible = false;
725      txtSearch.Text = "";
726      btnSearch.Visible = false;

```

```

727         mViewSearchResults.ActiveViewIndex = -1;
728
729         // Display the needed control.
730         switch ((Global.Enums.ItemTypes)ddlItemType.SelectedIndex)
731         {
732             case Global.Enums.ItemTypes.Business:
733                 ddlBusiness.Visible = true;
734                 break;
735
736             case Global.Enums.ItemTypes.Park:
737                 ddlPark.Visible = true;
738                 break;
739
740             case Global.Enums.ItemTypes.PublicFacility:
741                 ddlPublicFacility.Visible = true;
742                 break;
743
744             default:
745                 break;
746         }
747     }
748 }
749
750 /*-----
751 * Name:     ddlPark_SelectedIndexChanged
752 * Type:     Event Handler Method
753 * Purpose:  Handles the change of visibility on controls when the user chooses a park field.
754 * Input:    object sender, holds a reference to the object that raised this event.
755 * Input:    EventArgs e, holds data related to this event.
756 * Output:   Nothing.
757 -----*/
758 protected void ddlPark_SelectedIndexChanged(object sender, EventArgs e)
759 {
760     // Hide things until needed.
761     lblError.Visible = false;
762     lblResult.Visible = false;
763
764     ddlComparatorsStrings.Visible = false;
765     ddlComparatorsStrings.SelectedIndex = 0;
766     ddlComparatorsNotStrings.Visible = false;
767     ddlComparatorsNotStrings.SelectedIndex = 0;
768
769     txtSearch.Visible = false;
770     txtSearch.Text = "";
771     btnSearch.Visible = false;
772
773     mViewSearchResults.ActiveViewIndex = -1;
774
775     // Display the needed control.
776     switch ((Global.Enums.ParkFields)ddlPark.SelectedIndex)
777     {
778         case Global.Enums.ParkFields.Name:
779         case Global.Enums.ParkFields.Type:
780         case Global.Enums.ParkFields.StreetAddress:
781         case Global.Enums.ParkFields.City:
782         case Global.Enums.ParkFields.State:
783         case Global.Enums.ParkFields.Zip:
784         case Global.Enums.ParkFields.Phone:
785             ddlComparatorsStrings.Visible = true;
786             break;
787
788         case Global.Enums.ParkFields.Latitude:
789         case Global.Enums.ParkFields.Longitude:
790         case Global.Enums.ParkFields.FeatureBaseball:
791         case Global.Enums.ParkFields.FeatureBasketball:
792         case Global.Enums.ParkFields.FeatureGolf:

```

```

793         case Global.Enums.ParkFields.FeatureLargeMPField:
794         case Global.Enums.ParkFields.FeatureTennis:
795         case Global.Enums.ParkFields.FeatureVolleyball:
796             ddlComparatorsNotStrings.Visible = true;
797             break;
798
799         default:
800             break;
801     }
802 }
803
804 /*-----*/
805 * Name:     ddlPublicFacility_SelectedIndexChanged
806 * Type:     Event Handler Method
807 * Purpose:  Handles the change of visibility on controls when the user chooses a public
808 *           facility field.
809 * Input:    object sender, holds a reference to the object that raised this event.
810 * Input:    EventArgs e, holds data related to this event.
811 * Output:   Nothing.
812 /*-----*/
813 protected void ddlPublicFacility_SelectedIndexChanged(object sender, EventArgs e)
814 {
815     // Hide things until needed.
816     lblError.Visible = false;
817     lblResult.Visible = false;
818
819     ddlComparatorsStrings.Visible = false;
820     ddlComparatorsStrings.SelectedIndex = 0;
821     ddlComparatorsNotStrings.Visible = false;
822     ddlComparatorsNotStrings.SelectedIndex = 0;
823
824     txtSearch.Visible = false;
825     txtSearch.Text = "";
826     btnSearch.Visible = false;
827
828     mViewSearchResults.ActiveViewIndex = -1;
829
830     // Display the needed control.
831     switch ((Global.Enums.PublicFacilityFields)ddlPublicFacility.SelectedIndex)
832     {
833         case Global.Enums.PublicFacilityFields.Name:
834         case Global.Enums.PublicFacilityFields.Type:
835         case Global.Enums.PublicFacilityFields.StreetAddress:
836         case Global.Enums.PublicFacilityFields.City:
837         case Global.Enums.PublicFacilityFields.State:
838         case Global.Enums.PublicFacilityFields.Zip:
839         case Global.Enums.PublicFacilityFields.Phone:
840             ddlComparatorsStrings.Visible = true;
841             break;
842
843         case Global.Enums.PublicFacilityFields.Latitude:
844         case Global.Enums.PublicFacilityFields.Longitude:
845             ddlComparatorsNotStrings.Visible = true;
846             break;
847
848         default:
849             break;
850     }
851 }
852 }
853 }

```

```

1  /*-----
2  * Author:      Dan Cassidy
3  * Date:        2015-06-23
4  * Assignment:  cView-P4
5  * Source File: Statistics.aspx.cs
6  * Language:    C#
7  * Course:      CSCI-C 490, C# Programming, Mowe 08:00
8  * Purpose:     Code-behind file for Statistics.aspx. Controls the statistics displayed.
9  *-----*/
10
11 using System;
12 using System.Collections.Generic;
13 using System.Globalization;
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 Statistics : System.Web.UI.Page
22     {
23         /*-----
24         * Name:      Page_Load
25         * Type:      Event Handler Method
26         * Purpose:   Handles anything that should happen on page load. In this case, it calculates
27         *             and displays some statistics.
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 Page_Load(object sender, EventArgs e)
33         {
34             try
35             {
36                 using (CViewDataEntities database = new CViewDataEntities())
37                 {
38                     // Hijacked from http://stackoverflow.com/a/1206029.
39                     CultureInfo textInfo = new CultureInfo("en-US", false).TextInfo;
40
41                     // Total number of parks.
42                     lblStatistics1.Text = database.Parks.Count().ToString();
43
44                     // Total number of parks, grouped by park type.
45                     lblStatistics2.Text = "";
46                     var parksByType = database.Parks.GroupBy(p => p.Type);
47                     foreach (var parkType in parksByType)
48                         lblStatistics2.Text += (parkType.Key == "" ? "(Empty)" :
49                             textInfo.ToTitleCase(parkType.Key.ToLower())) + ": " +
50                             parkType.Count() + "<br />";
51
52                     // Total number of businesses.
53                     lblStatistics3.Text = database.Businesses.Count().ToString();
54
55                     // Total number of license renewals for each business.
56                     lblStatistics4.Text = "";
57                     var businessRenewals = database.Businesses.Where(
58                         b => b.LicenseStatus == "Renewed").GroupBy(b => b.Name);
59                     foreach (var business in businessRenewals)
60                         lblStatistics4.Text += (business.Key == "" ? "(Empty)" :
61                             textInfo.ToTitleCase(business.Key.ToLower())) + ": " +
62                             business.Count() + "<br />";
63
64                     // Total number of facilities that have the substring "Fire"
65                     lblStatistics5.Text = database.PublicFacilities.Where(pf => pf.Name.Contains(
66                         "fire")).Count().ToString();

```

```
67         }
68     }
69     catch
70     {
71         lblError.Text = "Error: Could not calculate statistics.";
72         lblError.Visible = true;
73     }
74 }
75 }
76 }
```



```

1  /*-----
2  * Author:      Dan Cassidy
3  * Date:        2015-06-23
4  * Assignment:  cView-P4
5  * Source File: CViewDataModelExtension.cs
6  * Language:    C#
7  * Course:      CSCI-C 490, C# Programming, MoWe 08:00
8  * Purpose:     Extends the Business, Park, and PublicFacility class generated by the ASP.NET Entity
9  *              Framework for this solution.
10 -----*/
11
12 using System;
13 using System.Collections.Generic;
14 using System.Linq;
15 using System.Web;
16
17 namespace cView_P4_DanCassidy
18 {
19     public partial class Business
20     {
21         /*-----
22         * Name:      this[]
23         * Type:      Indexer
24         * Purpose:   Provides easy access to the properties of the class.
25         * Input:     int fiendNum, represents the desired property.
26         * Output:    object, contains whichever property was desired, or null if the property was not
27         *              found.
28         -----*/
29         public object this[int fieldNum]
30         {
31             get
32             {
33                 switch ((Global.Enums.BusinessFields)fieldNum)
34                 {
35                     case Global.Enums.BusinessFields.Name:
36                         return Name;
37                     case Global.Enums.BusinessFields.Type:
38                         return Type;
39                     case Global.Enums.BusinessFields.StreetAddress:
40                         return StreetAddress;
41                     case Global.Enums.BusinessFields.City:
42                         return City;
43                     case Global.Enums.BusinessFields.State:
44                         return State;
45                     case Global.Enums.BusinessFields.Zip:
46                         return Zip;
47                     case Global.Enums.BusinessFields.Latitude:
48                         return Latitude;
49                     case Global.Enums.BusinessFields.Longitude:
50                         return Longitude;
51                     case Global.Enums.BusinessFields.Phone:
52                         return Phone;
53                     case Global.Enums.BusinessFields.LicenseNumber:
54                         return LicenseNumber;
55                     case Global.Enums.BusinessFields.LicenseIssueDate:
56                         return LicenseIssueDate;
57                     case Global.Enums.BusinessFields.LicenseExpirDate:
58                         return LicenseExpirDate;
59                     case Global.Enums.BusinessFields.LicenseStatus:
60                         return LicenseStatus;
61                     case Global.Enums.BusinessFields.CouncilDistrict:
62                         return CouncilDistrict;
63                     default:
64                         return null;
65                 }
66             }
67         }
68     }
69 }

```

```

67     }
68 }
69
70 public partial class Park
71 {
72     /*-----
73     * Name:      this[]
74     * Type:      Indexer
75     * Purpose:   Provides easy access to the properties of the class.
76     * Input:     int fiendNum, represents the desired property.
77     * Output:    object, contains whichever property was desired, or null if the property was not
78     *            found.
79     -----*/
80     public object this[int fieldNum]
81     {
82         get
83         {
84             switch ((Global.Enums.ParkFields)fieldNum)
85             {
86                 case Global.Enums.ParkFields.Name:
87                     return Name;
88                 case Global.Enums.ParkFields.Type:
89                     return Type;
90                 case Global.Enums.ParkFields.StreetAddress:
91                     return StreetAddress;
92                 case Global.Enums.ParkFields.City:
93                     return City;
94                 case Global.Enums.ParkFields.State:
95                     return State;
96                 case Global.Enums.ParkFields.Zip:
97                     return Zip;
98                 case Global.Enums.ParkFields.Latitude:
99                     return Latitude;
100                case Global.Enums.ParkFields.Longitude:
101                    return Longitude;
102                case Global.Enums.ParkFields.Phone:
103                    return Phone;
104                case Global.Enums.ParkFields.FeatureBaseball:
105                    return FeatureBaseball;
106                case Global.Enums.ParkFields.FeatureBasketball:
107                    return FeatureBasketball;
108                case Global.Enums.ParkFields.FeatureGolf:
109                    return FeatureGolf;
110                case Global.Enums.ParkFields.FeatureLargeMPField:
111                    return FeatureLargeMPField;
112                case Global.Enums.ParkFields.FeatureTennis:
113                    return FeatureTennis;
114                case Global.Enums.ParkFields.FeatureVolleyball:
115                    return FeatureVolleyball;
116                default:
117                    return null;
118            }
119        }
120    }
121 }
122
123 public partial class PublicFacility
124 {
125     /*-----
126     * Name:      this[]
127     * Type:      Indexer
128     * Purpose:   Provides easy access to the properties of the class.
129     * Input:     int fiendNum, represents the desired property.
130     * Output:    object, contains whichever property was desired, or null if the property was not
131     *            found.
132     -----*/

```

```
133     public object this[int fieldNum]
134     {
135         get
136         {
137             switch ((Global.Enums.PublicFacilityFields)fieldNum)
138             {
139                 case Global.Enums.PublicFacilityFields.Name:
140                     return Name;
141                 case Global.Enums.PublicFacilityFields.Type:
142                     return Type;
143                 case Global.Enums.PublicFacilityFields.StreetAddress:
144                     return StreetAddress;
145                 case Global.Enums.PublicFacilityFields.City:
146                     return City;
147                 case Global.Enums.PublicFacilityFields.State:
148                     return State;
149                 case Global.Enums.PublicFacilityFields.Zip:
150                     return Zip;
151                 case Global.Enums.PublicFacilityFields.Latitude:
152                     return Latitude;
153                 case Global.Enums.PublicFacilityFields.Longitude:
154                     return Longitude;
155                 case Global.Enums.PublicFacilityFields.Phone:
156                     return Phone;
157                 default:
158                     return null;
159             }
160         }
161     }
162 }
163 }
```

```

1  /*-----
2  * Author:      Dan Cassidy
3  * Date:        2015-06-23
4  * Assignment:  cView-P4
5  * Source File: SimpleConvert.cs
6  * Language:    C#
7  * Course:      CSCI-C 490, C# Programming, MoWe 08:00
8  * Purpose:     Provides simplified variants and extensions of some Convert methods.
9  *-----*/
10
11 namespace System
12 {
13     public static class SimpleConvert
14     {
15         /*-----
16         * Name:      ToByte
17         * Type:       Method
18         * Purpose:   Attempts to convert the given parameter, but returns the default object value if
19         *             it fails for any reason.
20         * Input:     string value, containing the value on which conversion will be attempted.
21         * Output:    byte object containing either the converted value or the default object value.
22         *-----*/
23         public static byte ToByte(string value)
24         {
25             try
26             {
27                 return Convert.ToByte(value);
28             }
29             catch
30             {
31                 return default(byte);
32             }
33         }
34
35         /*-----
36         * Name:      ToByteN
37         * Type:       Method
38         * Purpose:   Attempts to convert the given parameter, but returns a null if it fails for any
39         *             reason.
40         * Input:     string value, containing the value on which conversion will be attempted.
41         * Output:    Nullable<byte> object containing either the converted value or a null.
42         *-----*/
43         public static Nullable<byte> ToByteN(string value)
44         {
45             try
46             {
47                 return Convert.ToByte(value);
48             }
49             catch
50             {
51                 return null;
52             }
53         }
54
55         /*-----
56         * Name:      ToDateTime
57         * Type:       Method
58         * Purpose:   Attempts to convert the given parameter, but returns the default object value if
59         *             it fails for any reason.
60         * Input:     string value, containing the value on which conversion will be attempted.
61         * Output:    DateTime object containing either the converted value or the default object
62         *             value.
63         *-----*/
64         public static DateTime ToDateTime(string value)
65         {
66             try

```

```

67         {
68             return Convert.ToDateTime(value);
69         }
70     catch
71     {
72         return default(DateTime);
73     }
74 }
75
76 /*-----
77  * Name:    ToDateTimeN
78  * Type:    Method
79  * Purpose: Attempts to convert the given parameter, but returns a null if it fails for any
80  *           reason.
81  * Input:   string value, containing the value on which conversion will be attempted.
82  * Output:  Nullable<DateTime> object containing either the converted value or a null.
83  *-----*/
84 public static Nullable<DateTime> ToDateTimeN(string value)
85 {
86     try
87     {
88         return Convert.ToDateTime(value);
89     }
90     catch
91     {
92         return null;
93     }
94 }
95
96 /*-----
97  * Name:    ToDecimal
98  * Type:    Method
99  * Purpose: Attempts to convert the given parameter, but returns the default object value if
100  *           it fails for any reason.
101  * Input:   string value, containing the value on which conversion will be attempted.
102  * Output:  Decimal object containing either the converted value or the default object
103  *           value.
104  *-----*/
105 public static decimal ToDecimal(string value)
106 {
107     try
108     {
109         return Convert.ToDecimal(value);
110     }
111     catch
112     {
113         return default(decimal);
114     }
115 }
116
117 /*-----
118  * Name:    ToDecimalN
119  * Type:    Method
120  * Purpose: Attempts to convert the given parameter, but returns a null if it fails for any
121  *           reason.
122  * Input:   string value, containing the value on which conversion will be attempted.
123  * Output:  Nullable<Decimal> object containing either the converted value or a null.
124  *-----*/
125 public static Nullable<decimal> ToDecimalN(string value)
126 {
127     try
128     {
129         return Convert.ToDecimal(value);
130     }
131     catch
132     {

```

```

133         return null;
134     }
135 }
136
137 /*-----
138  * Name:    ToInt32
139  * Type:    Method
140  * Purpose: Attempts to convert the given parameter, but returns the default object value if
141  *           it fails for any reason.
142  * Input:   string value, containing the value on which conversion will be attempted.
143  * Output:  int object containing either the converted value or the default object value.
144  *-----*/
145 public static int ToInt32(string value)
146 {
147     try
148     {
149         return Convert.ToInt32(value);
150     }
151     catch
152     {
153         return default(int);
154     }
155 }
156
157 /*-----
158  * Name:    ToInt32N
159  * Type:    Method
160  * Purpose: Attempts to convert the given parameter, but returns a null if it fails for any
161  *           reason.
162  * Input:   string value, containing the value on which conversion will be attempted.
163  * Output:  Nullable<int> object containing either the converted value or a null.
164  *-----*/
165 public static Nullable<int> ToInt32N(string value)
166 {
167     try
168     {
169         return Convert.ToInt32(value);
170     }
171     catch
172     {
173         return null;
174     }
175 }
176
177 /*-----
178  * Name:    ToSingle
179  * Type:    Method
180  * Purpose: Attempts to convert the given parameter, but returns the default object value if
181  *           it fails for any reason.
182  * Input:   string value, containing the value on which conversion will be attempted.
183  * Output:  float object containing either the converted value or the default object value.
184  *-----*/
185 public static float ToSingle(string value)
186 {
187     try
188     {
189         return Convert.ToSingle(value);
190     }
191     catch
192     {
193         return default(float);
194     }
195 }
196
197 /*-----
198  * Name:    ToSingleN

```

```
199      * Type:      Method
200      * Purpose: Attempts to convert the given parameter, but returns the default object value if
201      *           it fails for any reason.
202      * Input:   string value, containing the value on which conversion will be attempted.
203      * Output:  Nullable<float> object containing either the converted value or a null.
204      -----*/
205      public static Nullable<float> ToSingleN(string value)
206      {
207          try
208          {
209              return Convert.ToSingle(value);
210          }
211          catch
212          {
213              return null;
214          }
215      }
216  }
217 }
218
```

```

1  /*-----
2  * Author:      Dan Cassidy
3  * Date:        2015-06-23
4  * Assignment:  cView-P4
5  * Source File: Global.asax.cs
6  * Language:    C#
7  * Course:      CSCI-C 490, C# Programming, MoWe 08:00
8  * Purpose:     Code-behind file for Global.asax. Contains things that should be globally
9  *              accessible, such as enums, exceptions, and strings.
10 -----*/
11
12 using System;
13 using System.Collections.Generic;
14 using System.Linq;
15 using System.Web;
16 using System.Web.Security;
17 using System.Web.SessionState;
18
19 namespace cView_P4_DanCassidy
20 {
21     public class Global : System.Web.HttpApplication
22     {
23         public static class Enums
24         {
25             /*-----
26             * Name:      BusinessFields
27             * Type:      Enum
28             * Purpose: Represents the different fields present in the Business class.
29             -----*/
30             public enum BusinessFields
31             {
32                 Name = 1,
33                 Type,
34                 StreetAddress,
35                 City,
36                 State,
37                 Zip,
38                 Latitude,
39                 Longitude,
40                 Phone,
41                 LicenseNumber,
42                 LicenseIssueDate,
43                 LicenseExpirDate,
44                 LicenseStatus,
45                 CouncilDistrict
46             }
47
48             /*-----
49             * Name:      ComparatorsNotStrings
50             * Type:      Enum
51             * Purpose: Represents the possible comparators available for use on non-strings.
52             -----*/
53             public enum ComparatorsNotStrings
54             {
55                 Contain = 1,
56                 NotContain,
57                 Equal,
58                 NotEqual,
59                 Greater,
60                 Less,
61                 GreaterEqual,
62                 LessEqual
63             }
64
65             /*-----
66             * Name:      ComparatorsStrings

```



```

67         * Type:      Enum
68         * Purpose: Represents the possible comparators available for use on strings.
69         -----*/
70     public enum ComparatorsStrings
71     {
72         Contain = 1,
73         NotContain,
74         Equal,
75         NotEqual
76     }
77
78     /*-----
79     * Name:      ItemTypes
80     * Type:      Enum
81     * Purpose: Represents the different types of items.
82     -----*/
83     public enum ItemTypes
84     {
85         Business = 1,
86         Park,
87         PublicFacility,
88     }
89
90     /*-----
91     * Name:      ParkFields
92     * Type:      Enum
93     * Purpose: Represents the different fields present in the Park class.
94     -----*/
95     public enum ParkFields
96     {
97         Name = 1,
98         Type,
99         StreetAddress,
100        City,
101        State,
102        Zip,
103        Latitude,
104        Longitude,
105        Phone,
106        FeatureBaseball,
107        FeatureBasketball,
108        FeatureGolf,
109        FeatureLargeMPField,
110        FeatureTennis,
111        FeatureVolleyball
112     }
113
114     /*-----
115     * Name:      PublicFacilityFields
116     * Type:      Enum
117     * Purpose: Represents the different fiels present in the Public Facility class.
118     -----*/
119     public enum PublicFacilityFields
120     {
121         Name = 1,
122         Type,
123         StreetAddress,
124         City,
125         State,
126         Zip,
127         Latitude,
128         Longitude,
129         Phone
130     }
131
132 }

```

```

133
134 public class Exceptions
135 {
136     /*-----
137     * Name: DuplicatePKException
138     * Type: Exception
139     * Purpose: Intended to describe a situation where an object with a duplicate primary
140     *           key attempted to be inserted into a primary keyed data structure.
141     -----*/
142     [Serializable]
143     public class DuplicatePKException : Exception
144     {
145         public DuplicatePKException() { }
146
147         public DuplicatePKException(string message)
148             : base(message) { }
149
150         public DuplicatePKException(string keyName, object keyValue)
151             : base(string.Format("An item already exists with a {0} of {1}.",
152             keyName, keyValue)) { }
153
154         public DuplicatePKException(string message, Exception inner)
155             : base(message, inner) { }
156
157         protected DuplicatePKException(
158             System.Runtime.Serialization.SerializationInfo info,
159             System.Runtime.Serialization.StreamingContext context)
160             : base(info, context) { }
161     }
162
163     /*-----
164     * Name: EmptyOrNullPKException
165     * Type: Exception
166     * Purpose: Intended to describe a situation where an object with an empty or null
167     *           primary key attempted to be inserted into a primary keyed data structure.
168     -----*/
169     [Serializable]
170     public class EmptyOrNullPKException : Exception
171     {
172         public EmptyOrNullPKException() { }
173
174         public EmptyOrNullPKException(string keyName)
175             : base(string.Format("{0} cannot be empty or null.", keyName)) { }
176
177         public EmptyOrNullPKException(string message, Exception inner)
178             : base(message, inner) { }
179
180         protected EmptyOrNullPKException(
181             System.Runtime.Serialization.SerializationInfo info,
182             System.Runtime.Serialization.StreamingContext context)
183             : base(info, context) { }
184     }
185 }
186
187     /*-----
188     * Name: Strings
189     * Type: Class
190     * Purpose: Contains common strings used throughout the application in a centrally-managed
191     *           location.
192     -----*/
193     public static class Strings
194     {
195         public const string Separator = ":";
196
197         public const string BusinessName = "Business Name";
198         public const string ParkName = "Park Name";

```

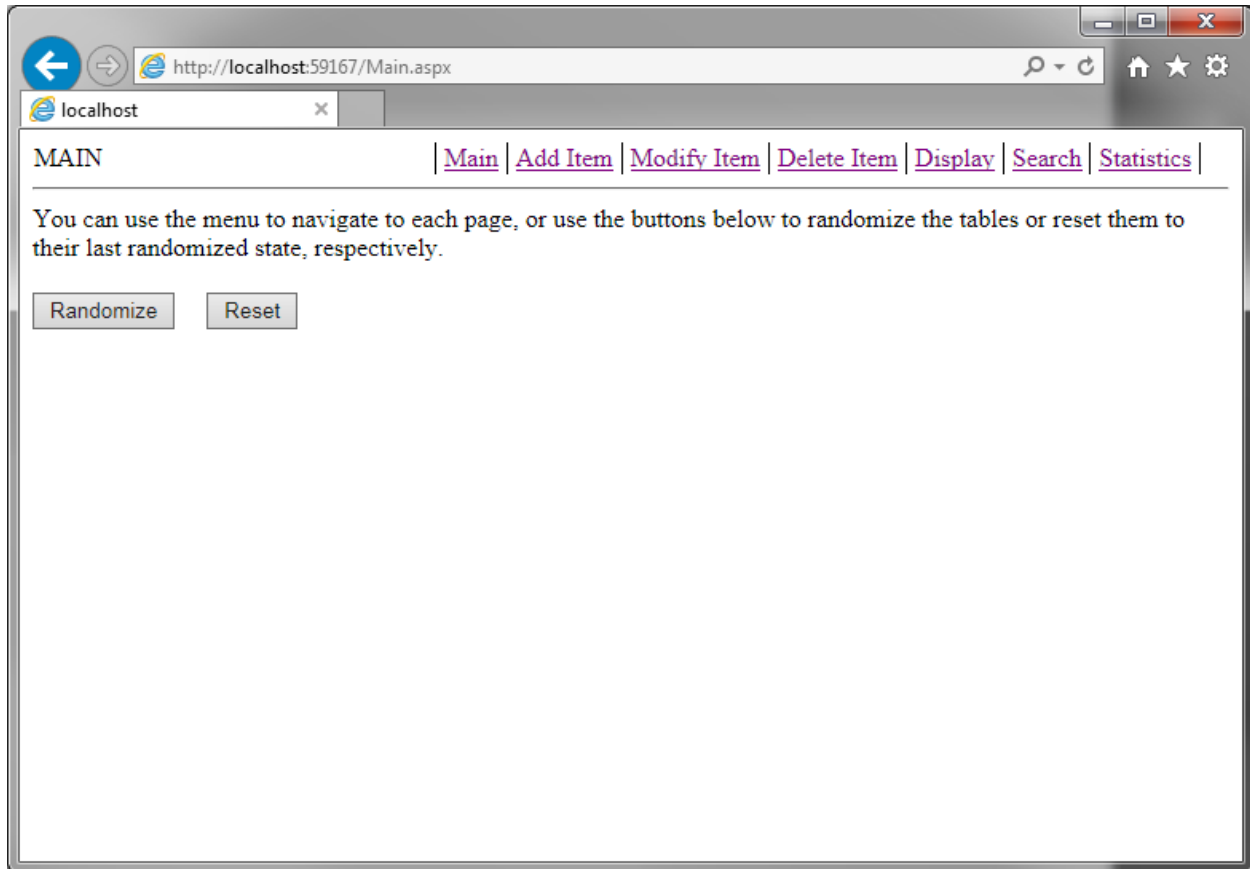
```
199         public const string PublicFacilityName = "Public Facility Name";
200
201         public const string BusinessType = "Type of Business";
202         public const string ParkType = "Type of Park";
203         public const string PublicFacilityType = "Type of Public Facility";
204
205         public const string BusinessKey = "License Number";
206         public const string ParkKey = ParkName;
207         public const string PublicFacilityKey = PublicFacilityName;
208     }
209
210 }
211 }
```

```
1  -- NOTE:      The other two tables I used that dealt with business objects,
2  --            BusinessBase and BusinessReset are created in the exact same
3  --            manner (allowing for differing table names and constraint names,
4  --            so I am omitting their SQL statements for the sake of brevity.
5  USE [djccassid]
6  GO
7
8  SET ANSI_NULLS ON
9  GO
10
11 SET QUOTED_IDENTIFIER ON
12 GO
13
14 SET ANSI_PADDING ON
15 GO
16
17 CREATE TABLE [dbo].[Business](
18     [Name] [varchar](100) NOT NULL,
19     [Type] [varchar](50) NOT NULL,
20     [StreetAddress] [varchar](50) NOT NULL,
21     [City] [varchar](50) NOT NULL,
22     [State] [varchar](50) NOT NULL,
23     [Zip] [varchar](50) NOT NULL,
24     [Latitude] [decimal](18, 15) NOT NULL,
25     [Longitude] [decimal](18, 15) NOT NULL,
26     [Phone] [varchar](50) NOT NULL,
27     [LicenseNumber] [varchar](50) NOT NULL,
28     [LicenseIssueDate] [date] NOT NULL,
29     [LicenseExpirDate] [date] NOT NULL,
30     [LicenseStatus] [varchar](50) NOT NULL,
31     [CouncilDistrict] [varchar](50) NOT NULL,
32     CONSTRAINT [PK_Business_LicenseNumber] PRIMARY KEY CLUSTERED
33     (
34         [LicenseNumber] ASC
35     )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
36     ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
37     ) ON [PRIMARY]
38 GO
39
40 SET ANSI_PADDING OFF
41 GO
42
43 ALTER TABLE [dbo].[Business] ADD CONSTRAINT [DF_Business_Latitude] DEFAULT ((0)) FOR [Latitude]
44 GO
45
46 ALTER TABLE [dbo].[Business] ADD CONSTRAINT [DF_Business_Longitude] DEFAULT ((0)) FOR [Longitude]
47 GO
48
49 ALTER TABLE [dbo].[Business] ADD CONSTRAINT [DF_Business_LicenseIssueDate] DEFAULT ('0001-01-01') FOR
50 [LicenseIssueDate]
51 GO
52 ALTER TABLE [dbo].[Business] ADD CONSTRAINT [DF_Business_LicenseExpirDate] DEFAULT ('0001-01-01') FOR
53 [LicenseExpirDate]
54 GO
55
56
57 -- NOTE:      The other two tables I used that dealt with park objects,
58 --            ParkBase and ParkReset are created in the exact same
59 --            manner (allowing for differing table names and constraint names,
60 --            so I am omitting their SQL statements for the sake of brevity.
61 USE [djccassid]
```

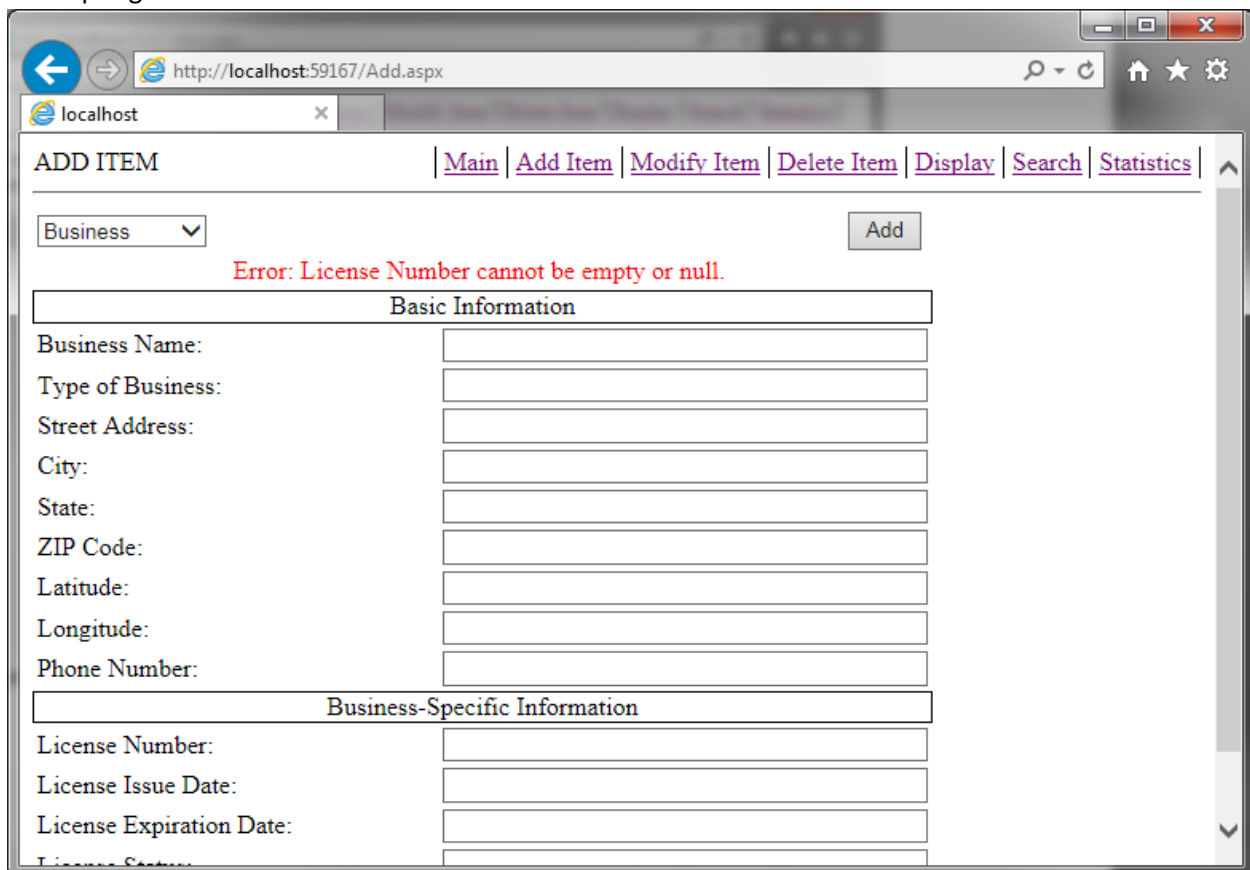
```
62 GO
63
64 SET ANSI_NULLS ON
65 GO
66
67 SET QUOTED_IDENTIFIER ON
68 GO
69
70 SET ANSI_PADDING ON
71 GO
72
73 CREATE TABLE [dbo].[Park](
74     [Name] [varchar](100) NOT NULL,
75     [Type] [varchar](50) NOT NULL,
76     [StreetAddress] [varchar](50) NOT NULL,
77     [City] [varchar](50) NOT NULL,
78     [State] [varchar](50) NOT NULL,
79     [Zip] [varchar](50) NOT NULL,
80     [Latitude] [decimal](18, 15) NOT NULL,
81     [Longitude] [decimal](18, 15) NOT NULL,
82     [Phone] [varchar](50) NOT NULL,
83     [FeatureBaseball] [tinyint] NOT NULL,
84     [FeatureBasketball] [decimal](4, 1) NOT NULL,
85     [FeatureGolf] [decimal](4, 1) NOT NULL,
86     [FeatureLargeMPField] [tinyint] NOT NULL,
87     [FeatureTennis] [tinyint] NOT NULL,
88     [FeatureVolleyball] [tinyint] NOT NULL,
89     CONSTRAINT [PK_Park_Name] PRIMARY KEY CLUSTERED
90 (
91     [Name] ASC
92 )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
93     ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
94 ) ON [PRIMARY]
95 GO
96
97 SET ANSI_PADDING OFF
98 GO
99 ALTER TABLE [dbo].[Park] ADD CONSTRAINT [DF_Park_Latitude] DEFAULT ((0)) FOR [Latitude]
100 GO
101
102 ALTER TABLE [dbo].[Park] ADD CONSTRAINT [DF_Park_Longitude] DEFAULT ((0)) FOR [Longitude]
103 GO
104
105 ALTER TABLE [dbo].[Park] ADD CONSTRAINT [DF_Park_FeatureBaseball] DEFAULT ((0)) FOR [FeatureBaseball]
106 GO
107
108 ALTER TABLE [dbo].[Park] ADD CONSTRAINT [DF_Park_FeatureBasketball] DEFAULT ((0)) FOR [FeatureBasketball]
109 GO
110
111 ALTER TABLE [dbo].[Park] ADD CONSTRAINT [DF_Park_FeatureGolf] DEFAULT ((0)) FOR [FeatureGolf]
112 GO
113
114 ALTER TABLE [dbo].[Park] ADD CONSTRAINT [DF_Park_FeatureLargeMPField] DEFAULT ((0)) FOR [FeatureLargeMPField]
115 GO
116
117 ALTER TABLE [dbo].[Park] ADD CONSTRAINT [DF_Park_FeatureTennis] DEFAULT ((0)) FOR [FeatureTennis]
118 GO
119
120 ALTER TABLE [dbo].[Park] ADD CONSTRAINT [DF_Park_FeatureVolleyball] DEFAULT ((0)) FOR [FeatureVolleyball]
121 GO
122
123
124
```

```
125 -- NOTE:      The other two tables I used that dealt with public facility objects,
126 --            PublicFacilityBase and PublicFacilityReset are created in the exact same
127 --            manner (allowing for differing table names and constraint names,
128 --            so I am omitting their SQL statements for the sake of brevity.
129 USE [djccassid]
130 GO
131
132 SET ANSI_NULLS ON
133 GO
134
135 SET QUOTED_IDENTIFIER ON
136 GO
137
138 SET ANSI_PADDING ON
139 GO
140
141 CREATE TABLE [dbo].[PublicFacility](
142     [Name] [varchar](100) NOT NULL,
143     [Type] [varchar](50) NOT NULL,
144     [StreetAddress] [varchar](50) NOT NULL,
145     [City] [varchar](50) NOT NULL,
146     [State] [varchar](50) NOT NULL,
147     [Zip] [varchar](50) NOT NULL,
148     [Latitude] [decimal](18, 15) NOT NULL,
149     [Longitude] [decimal](18, 15) NOT NULL,
150     [Phone] [varchar](50) NOT NULL,
151     CONSTRAINT [PK_PublicFacility_Name] PRIMARY KEY CLUSTERED
152     (
153         [Name] ASC
154     )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
155     ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
156 ) ON [PRIMARY]
157 GO
158
159 SET ANSI_PADDING OFF
160 GO
161
162 ALTER TABLE [dbo].[PublicFacility] ADD CONSTRAINT [DF_PublicFacility_Latitude] DEFAULT ((0)) FOR [Latitude]
163 GO
164
165 ALTER TABLE [dbo].[PublicFacility] ADD CONSTRAINT [DF_PublicFacility_Longitude] DEFAULT ((0)) FOR [Longitude]
166 GO
167
```

Main/Menu page.



Attempting to add a business without a license number



Attempting to add a business with a license number that already exists

The screenshot shows a web browser window with the URL <http://localhost:59167/Add.aspx>. The page title is "ADD ITEM". There are navigation links: [Main](#), [Add Item](#), [Modify Item](#), [Delete Item](#), [Display](#), [Search](#), and [Statistics](#). A dropdown menu is set to "Business" and an "Add" button is visible. A red error message states: "Error: An item already exists with a License Number of 2006-727." Below the error message, there are two sections: "Basic Information" and "Business-Specific Information".

Basic Information	
Business Name:	sdkih
Type of Business:	sdkjg
Street Address:	
City:	
State:	
ZIP Code:	
Latitude:	
Longitude:	
Phone Number:	

Business-Specific Information	
License Number:	2006-727
License Issue Date:	
License Expiration Date:	
License Status:	

Attempting to add a park without a name

The screenshot shows a web browser window with the URL <http://localhost:59167/Add.aspx>. The page title is "ADD ITEM". There are navigation links: [Main](#), [Add Item](#), [Modify Item](#), [Delete Item](#), [Display](#), [Search](#), and [Statistics](#). A dropdown menu is set to "Park" and an "Add" button is visible. A red error message states: "Error: Park Name cannot be empty or null." Below the error message, there are two sections: "Basic Information" and "Park-Specific Information".

Basic Information	
Park Name:	
Type of Park:	
Street Address:	
City:	
State:	
ZIP Code:	
Latitude:	
Longitude:	
Phone Number:	

Park-Specific Information	
Number of Baseball Diamonds:	
Number of Basketball Courts:	
Number of Golf Courses:	
Number of Large Multipurpose Fields:	



Attempting to add a park with name that already exists

The screenshot shows a web browser window at <http://localhost:59167/Add.aspx>. The page has a navigation bar with links: [Main](#), [Add Item](#), [Modify Item](#), [Delete Item](#), [Display](#), [Search](#), and [Statistics](#). Below the navigation bar, there is a dropdown menu labeled "Park" and an "Add" button. A red error message is displayed: "Error: An item already exists with a Park Name of Belleville Park." Below the error message, there are two sections: "Basic Information" and "Park-Specific Information". The "Basic Information" section contains fields for: Park Name (filled with "Belleville Park"), Type of Park, Street Address, City, State, ZIP Code, Latitude, Longitude, and Phone Number. The "Park-Specific Information" section contains fields for: Number of Baseball Diamonds, Number of Basketball Courts, Number of Golf Courses, and Number of Large Multipurpose Fields.

Attempting to add a public facility without a name

The screenshot shows a web browser window at <http://localhost:59167/Add.aspx>. The page has a navigation bar with links: [Main](#), [Add Item](#), [Modify Item](#), [Delete Item](#), [Display](#), [Search](#), and [Statistics](#). Below the navigation bar, there is a dropdown menu labeled "Public Facility" and an "Add" button. A red error message is displayed: "Error: Public Facility Name cannot be empty or null." Below the error message, there is a section titled "Basic Information" containing fields for: Public Facility Name, Type of Public Facility, Street Address, City, State, ZIP Code, Latitude, Longitude, and Phone Number.

Attempting to add a public facility that already exists

The screenshot shows a web browser window with the URL <http://localhost:59167/Add.aspx>. The page title is "ADD ITEM". There are navigation links: [Main](#), [Add Item](#), [Modify Item](#), [Delete Item](#), [Display](#), [Search](#), and [Statistics](#). A dropdown menu is set to "Public Facility". An "Add" button is visible. A red error message states: "Error: An item already exists with a Public Facility Name of Mishawaka EMS Station." Below the error, a form titled "Basic Information" contains the following fields:

Public Facility Name:	Mishawaka EMS Station
Type of Public Facility:	
Street Address:	
City:	
State:	
ZIP Code:	
Latitude:	
Longitude:	
Phone Number:	

Successfully adding a business

The screenshot shows the same web browser window. The dropdown menu is now set to "Business". The "Add" button is visible. A green message states: "Added record to the table." Below the message, a form titled "Basic Information" contains the following fields:

Business Name:	Test Business
Type of Business:	Test
Street Address:	Test
City:	Test
State:	Test
ZIP Code:	Test
Latitude:	Test
Longitude:	Test
Phone Number:	Test

Below the "Basic Information" section is a section titled "Business-Specific Information" with the following fields:

License Number:	2001-1
License Issue Date:	
License Expiration Date:	
License Status:	

### Successfully adding a park

The screenshot shows a web browser window at <http://localhost:59167/Add.aspx>. The page has a navigation bar with links: [Main](#), [Add Item](#), [Modify Item](#), [Delete Item](#), [Display](#), [Search](#), and [Statistics](#). Below the navigation bar, there is a dropdown menu set to "Park" and an "Add" button. A message "Added record to the table." is displayed. The form is divided into two sections: "Basic Information" and "Park-Specific Information".

Basic Information	
Park Name:	Test Park
Type of Park:	Test
Street Address:	Test
City:	Test
State:	Test
ZIP Code:	Test
Latitude:	Test
Longitude:	Test
Phone Number:	Test

Park-Specific Information	
Number of Baseball Diamonds:	
Number of Basketball Courts:	
Number of Golf Courses:	
Number of Large Multipurpose Fields:	

### Successfully adding a public facility

The screenshot shows the same web browser window at <http://localhost:59167/Add.aspx>. The navigation bar and message "Added record to the table." are identical to the previous screenshot. The dropdown menu is now set to "Public Facility". The form is divided into two sections: "Basic Information" and "Park-Specific Information".

Basic Information	
Public Facility Name:	Test Public Facility
Type of Public Facility:	Test
Street Address:	Test
City:	Test
State:	Test
ZIP Code:	Test
Latitude:	Test
Longitude:	Test
Phone Number:	Test

Park-Specific Information	
Number of Baseball Diamonds:	
Number of Basketball Courts:	
Number of Golf Courses:	
Number of Large Multipurpose Fields:	

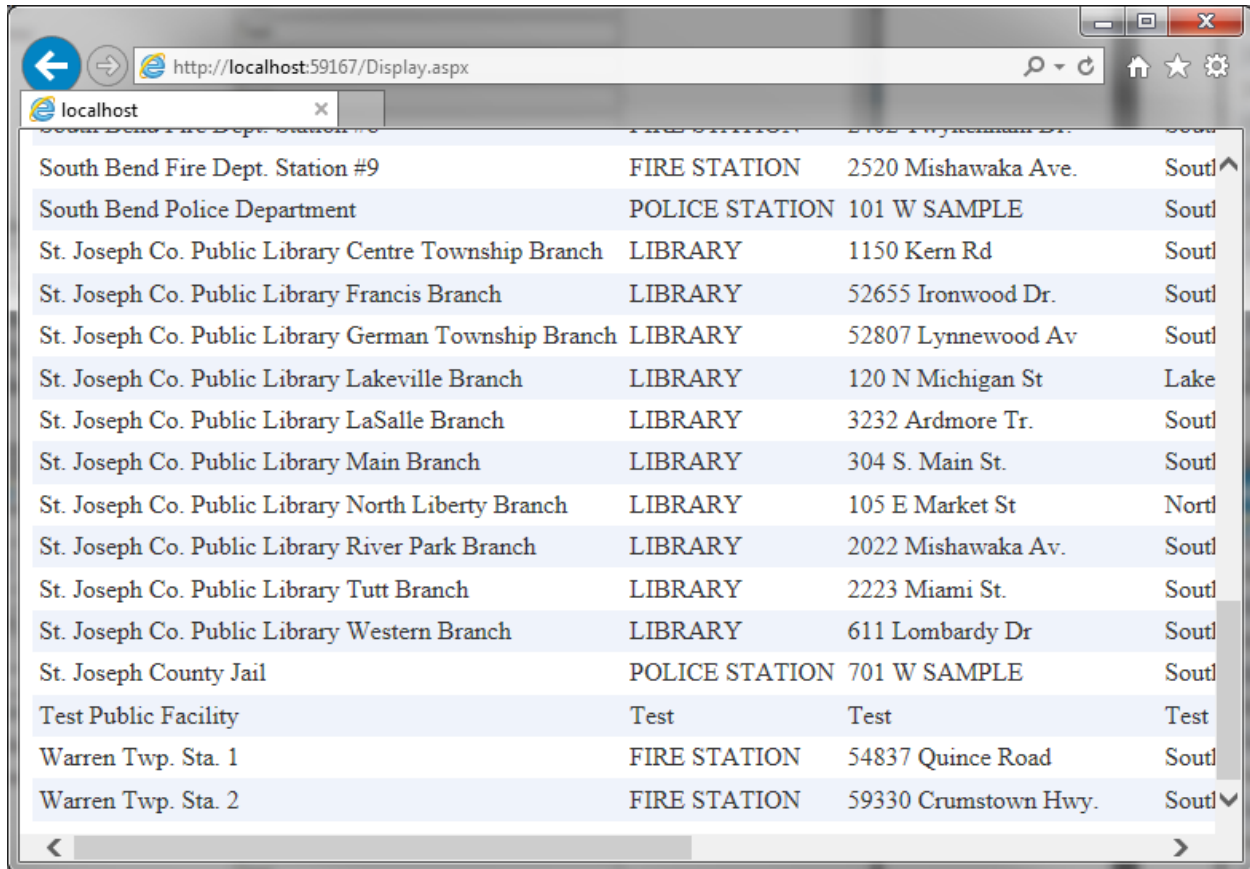
Displaying all businesses while noting that the test business is there

License Number	Business Name	Business Type	Street
2001-1	Test Business	Test	Test
2006-727	LOGAN'S ROADHOUSE	RESTAURANTS A-M	1320 E IRELAND
2006-735	HOWARD PARK GENERAL STORE	RESTAURANTS A-M	609 E JEFFERSON
2006-752	JULIO'S PIZZA	RESTAURANTS A-M	2509 S MICHIGAN
2007-1017	PIZZA HUT - DONT USE	RESTAURANTS N-Z	2017 SOUTH BEND
2007-1420	INN ON HISTORIC W WASHINGTON	RESTAURANTS A-M	322 W WASHINGTON
2008-2578	MTAMBO MBARAKA K	TAXI DRIVER	2309 LINCOLN
2008-2660	JALISCO SUPERMARKET	RESTAURANTS A-M	454 S CARLISLE
2008-2822	TONI BERRY	LAWN PARK UNDER 10 SPACES	434 NAPOLEON
2008-705	CHICORY CAFE CO. - DO NOT USE	RESTAURANTS A-M	105 E JEFFERSON
2009-1331	CHINA GARDEN RESTAURANT INC	RESTAURANTS A-M	900 E IRELAND
2009-3065	UNITED CAR	TAXI COMPANY	2730 MISHAWA

Displaying all parks while noting that the test park is there

Park Name	Block Park	Address	City	State	Zip
Ravina Park	Block Park	1522 Lebanon St.	South Bend	Indiana	46613 41
River Manor Park	Neighborhood Park	3811 Glenview Dr.	South Bend	Indiana	46628 41
Riverside Park	Neighborhood Park	3840 Riverside Drive	South Bend	Indiana	46628 41
Rum Village Annex	Community Park	2700 Gertrude St	South Bend	Indiana	46614 41
Ryer Mini Park	Block Park	1565 Ryer St.	South Bend	Indiana	46628 41
Seitz Park	Block Park	340 E. Colfax	South Bend	Indiana	46601 41
Sorin Mini Park	Block Park	3366 Sorin Street	South Bend	Indiana	46615 41
Southeast Neighborhood Park	Neighborhood Park	503 Wenger Street	South Bend	Indiana	46601 41
Studebaker Golf Course	Golf Course	718 Calvert Street	South Bend	Indiana	46613 41
Test Park	Test	Test	Test	Test	Test 0.0
Vassar Park	Neighborhood Park	1100 Sherman Avenue	South Bend	Indiana	46616 41
Viewing Park	Neighborhood Park	900 Northside Blvd	South Bend	Indiana	46617 41
Voorde Park	Neighborhood Park	3433 Keller St.	South Bend	Indiana	46628 41
Walker Field Park	Neighborhood Park	2212 S. Walnut Street	South Bend	Indiana	46613 41
Westhaven Park	Block Park	1220 Sussex Drive	South Bend	Indiana	46628 41
Woodlawn Park	Neighborhood Park	2166 Riverside Drive	South Bend	Indiana	46616 41

Displaying all public facilities while noting that the test public facility is there



South Bend Fire Dept. Station #9	FIRE STATION	2520 Mishawaka Ave.	South
South Bend Police Department	POLICE STATION	101 W SAMPLE	South
St. Joseph Co. Public Library Centre Township Branch	LIBRARY	1150 Kern Rd	South
St. Joseph Co. Public Library Francis Branch	LIBRARY	52655 Ironwood Dr.	South
St. Joseph Co. Public Library German Township Branch	LIBRARY	52807 Lynnewood Av	South
St. Joseph Co. Public Library Lakeville Branch	LIBRARY	120 N Michigan St	Lake
St. Joseph Co. Public Library LaSalle Branch	LIBRARY	3232 Ardmore Tr.	South
St. Joseph Co. Public Library Main Branch	LIBRARY	304 S. Main St.	South
St. Joseph Co. Public Library North Liberty Branch	LIBRARY	105 E Market St	North
St. Joseph Co. Public Library River Park Branch	LIBRARY	2022 Mishawaka Av.	South
St. Joseph Co. Public Library Tutt Branch	LIBRARY	2223 Miami St.	South
St. Joseph Co. Public Library Western Branch	LIBRARY	611 Lombardy Dr	South
St. Joseph County Jail	POLICE STATION	701 W SAMPLE	South
Test Public Facility	Test	Test	Test
Warren Twp. Sta. 1	FIRE STATION	54837 Quince Road	South
Warren Twp. Sta. 2	FIRE STATION	59330 Crumstown Hwy.	South

Selecting the test business for modification



MODIFY ITEM | [Main](#) | [Add Item](#) | [Modify Item](#) | [Delete Item](#) | [Display](#) | [Search](#) | [Statistics](#)

Business

Select	License Number	Business Name	Street Address
Select	2001-1	Test Business	Test
Select	2006-727	LOGAN'S ROADHOUSE	1320 E IRELAND RD
Select	2006-735	HOWARD PARK GENERAL STORE	609 E JEFFERSON BLVD
Select	2006-752	JULIO'S PIZZA	2509 S MICHIGAN ST
Select	2007-1017	PIZZA HUT - DONT USE	2017 SOUTH BEND AVE
Select	2007-1420	INN ON HISTORIC W WASHINGTON	322 W WASHINGTON ST
Select	2008-2578	MTAMBO MBARAKA K	2309 LINCOLN WAY W
Select	2008-2660	JALISCO SUPERMARKET	454 S CARLISLE ST
Select	2008-2822	TONI BERRY	434 NAPOLEON ST
Select	2008-705	CHICORY CAFE CO. - DO NOT USE	105 E JEFFERSON BLVD
Select	2009-1331	CHINA GARDEN RESTAURANT INC	900 E IRELAND RD
Select	2009-3065	UNITED CAB	2730 MISHAWAKA AVE
Select	2009-3137	ABC CAB & COURIER EXPRESS CAB	1733 S MICHIGAN ST

Observing that the text boxes are filled with the information for the test business

The screenshot shows a web browser window at <http://localhost:59167/Modify.aspx>. The page title is 'MODIFY ITEM'. At the top, there is a navigation bar with links: [Main](#), [Add Item](#), [Modify Item](#), [Delete Item](#), [Display](#), [Search](#), and [Statistics](#). Below the navigation bar, there is a 'Business' dropdown menu, a 'Back' button, and a 'Save Changes' button. The form is divided into two sections: 'Basic Information' and 'Business-Specific Information'. The 'Basic Information' section contains the following fields: Business Name (Test Business), Type of Business (Test), Street Address (Test), City (Test), State (Test), ZIP Code (Test), Latitude (0.0000000000000000), Longitude (0.0000000000000000), and Phone Number (Test). The 'Business-Specific Information' section contains the following fields: License Number (2001-1), License Issue Date (0001-01-01), License Expiration Date (0001-01-01), and License Status (empty).

Basic Information	
Business Name:	Test Business
Type of Business:	Test
Street Address:	Test
City:	Test
State:	Test
ZIP Code:	Test
Latitude:	0.0000000000000000
Longitude:	0.0000000000000000
Phone Number:	Test

Business-Specific Information	
License Number:	2001-1
License Issue Date:	0001-01-01
License Expiration Date:	0001-01-01
License Status:	

Showing the modification to be done to the test business

The screenshot shows the same web browser window at <http://localhost:59167/Modify.aspx>. The page title is 'MODIFY ITEM'. The navigation bar and buttons are the same as in the previous screenshot. The 'Business' dropdown menu is still set to 'Business'. The 'Basic Information' section contains the following fields: Business Name (Further Test Business), Type of Business (Test), Street Address (Test), City (Test), State (Test), ZIP Code (Test), Latitude (0.0000000000000000), Longitude (0.0000000000000000), and Phone Number (Test). The 'Business-Specific Information' section contains the following fields: License Number (2001-1), License Issue Date (0001-01-01), License Expiration Date (0001-01-01), and License Status (empty).

Basic Information	
Business Name:	Further Test Business
Type of Business:	Test
Street Address:	Test
City:	Test
State:	Test
ZIP Code:	Test
Latitude:	0.0000000000000000
Longitude:	0.0000000000000000
Phone Number:	Test

Business-Specific Information	
License Number:	2001-1
License Issue Date:	0001-01-01
License Expiration Date:	0001-01-01
License Status:	



Showing the completed modification to the test business

MODIFY ITEM | [Main](#) | [Add Item](#) | [Modify Item](#) | [Delete Item](#) | [Display](#) | [Search](#) | [Statistics](#)

Business

Item modified successfully.

<a href="#">Select</a>	<u>License Number</u>	<u>Business Name</u>	<u>Street Address</u>
<a href="#">Select</a>	2001-1	Further Test Business	Test
<a href="#">Select</a>	2006-727	LOGAN'S ROADHOUSE	1320 E IRELAND RD
<a href="#">Select</a>	2006-735	HOWARD PARK GENERAL STORE	609 E JEFFERSON BLVD
<a href="#">Select</a>	2006-752	JULIO'S PIZZA	2509 S MICHIGAN ST
<a href="#">Select</a>	2007-1017	PIZZA HUT - DONT USE	2017 SOUTH BEND AVE
<a href="#">Select</a>	2007-1420	INN ON HISTORIC W WASHINGTON	322 W WASHINGTON ST
<a href="#">Select</a>	2008-2578	MTAMBO MBARAKA K	2309 LINCOLN WAY W
<a href="#">Select</a>	2008-2660	JALISCO SUPERMARKET	454 S CARLISLE ST
<a href="#">Select</a>	2008-2822	TONI BERRY	434 NAPOLEON ST
<a href="#">Select</a>	2008-705	CHICORY CAFE CO. - DO NOT USE	105 E JEFFERSON BLVD
<a href="#">Select</a>	2009-1331	CHINA GARDEN RESTAURANT INC	900 E IRELAND RD
<a href="#">Select</a>	2009-3065	UNITED CAB	2730 MISHAWAKA AVE

Selecting the test park for modification

[Select](#) Randolph Mini Park 1809 Randolph Street

[Select](#) Ravina Park 1522 Lebanon St.

[Select](#) River Manor Park 3811 Glenview Dr.

[Select](#) Riverside Park 3840 Riverside Drive

[Select](#) Rum Village Annex 2700 Gertrude St

[Select](#) Ryer Mini Park 1565 Ryer St.

[Select](#) Seitz Park 340 E. Colfax

[Select](#) Sorin Mini Park 3366 Sorin Street

[Select](#) Southeast Neighborhood Park 503 Wenger Street

[Select](#) Studebaker Golf Course 718 Calvert Street

[Select](#) **Test Park** **Test**

[Select](#) Vassar Park 1100 Sherman Avenue

[Select](#) Viewing Park 900 Northside Blvd

[Select](#) Voorde Park 3433 Keller St.

[Select](#) Walker Field Park 2212 S. Walnut Street

[Select](#) Westhaven Park 1220 Sussex Drive

[Select](#) Woodlawn Park 2166 Riverside Drive

Observing that the text boxes are filled with the information for the test park

The screenshot shows a web browser window at <http://localhost:59167/Modify.aspx>. The page title is 'MODIFY ITEM'. At the top, there is a navigation bar with links: [Main](#), [Add Item](#), [Modify Item](#), [Delete Item](#), [Display](#), [Search](#), and [Statistics](#). Below the navigation bar, there is a dropdown menu labeled 'Park' and two buttons: 'Back' and 'Save Changes'. The form is divided into two sections: 'Basic Information' and 'Park-Specific Information'. The 'Basic Information' section contains the following fields: 'Park Name' (Test Park), 'Type of Park' (Test), 'Street Address' (Test), 'City' (Test), 'State' (Test), 'ZIP Code' (Test), 'Latitude' (0.0000000000000000), 'Longitude' (0.0000000000000000), and 'Phone Number' (Test). The 'Park-Specific Information' section contains the following fields: 'Number of Baseball Diamonds' (0), 'Number of Basketball Courts' (0.0), 'Number of Golf Courses' (0.0), and 'Number of Large Multipurpose Fields' (0).

Basic Information	
Park Name:	Test Park
Type of Park:	Test
Street Address:	Test
City:	Test
State:	Test
ZIP Code:	Test
Latitude:	0.0000000000000000
Longitude:	0.0000000000000000
Phone Number:	Test

Park-Specific Information	
Number of Baseball Diamonds:	0
Number of Basketball Courts:	0.0
Number of Golf Courses:	0.0
Number of Large Multipurpose Fields:	0

Showing the modification to be done to the test park

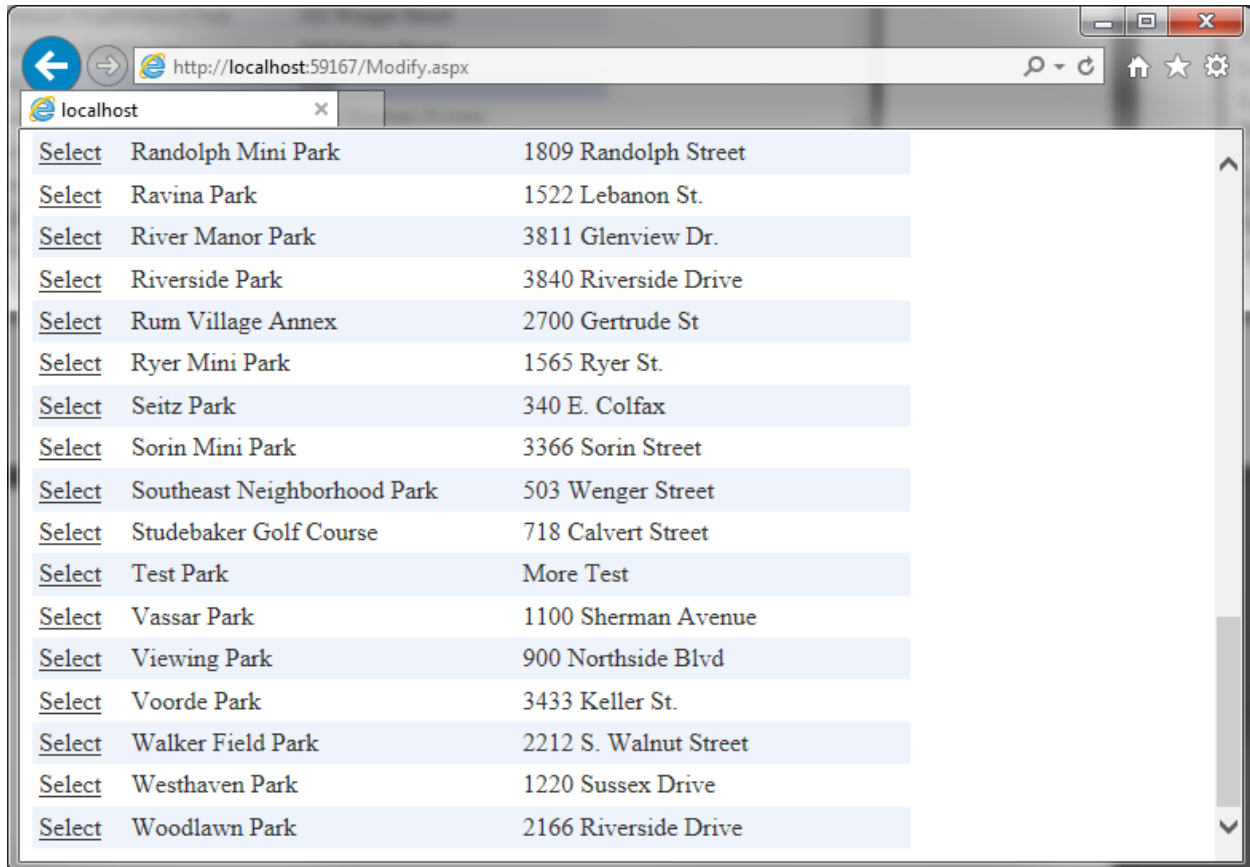
The screenshot shows the same web browser window as the previous one, but with a modification to the 'Street Address' field. The 'Street Address' field now contains 'More Test' and has a small 'x' icon next to it, indicating a change. All other fields remain the same as in the previous screenshot.

Basic Information	
Park Name:	Test Park
Type of Park:	Test
Street Address:	More Test
City:	Test
State:	Test
ZIP Code:	Test
Latitude:	0.0000000000000000
Longitude:	0.0000000000000000
Phone Number:	Test

Park-Specific Information	
Number of Baseball Diamonds:	0
Number of Basketball Courts:	0.0
Number of Golf Courses:	0.0
Number of Large Multipurpose Fields:	0




Showing the completed modification to the test park



A screenshot of a web browser window displaying a list of parks. The browser's address bar shows 'http://localhost:59167/Modify.aspx'. The list contains 15 items, each with a 'Select' link, a park name, and an address. The 'Test Park' entry is highlighted in blue.

Select	Randolph Mini Park	1809 Randolph Street
Select	Ravina Park	1522 Lebanon St.
Select	River Manor Park	3811 Glenview Dr.
Select	Riverside Park	3840 Riverside Drive
Select	Rum Village Annex	2700 Gertrude St
Select	Ryer Mini Park	1565 Ryer St.
Select	Seitz Park	340 E. Colfax
Select	Sorin Mini Park	3366 Sorin Street
Select	Southeast Neighborhood Park	503 Wenger Street
Select	Studebaker Golf Course	718 Calvert Street
Select	Test Park	More Test
Select	Vassar Park	1100 Sherman Avenue
Select	Viewing Park	900 Northside Blvd
Select	Voorde Park	3433 Keller St.
Select	Walker Field Park	2212 S. Walnut Street
Select	Westhaven Park	1220 Sussex Drive
Select	Woodlawn Park	2166 Riverside Drive

Selecting the test public facility for modification



A screenshot of a web browser window displaying a list of public facilities. The browser's address bar shows 'http://localhost:59167/Modify.aspx'. The list contains 18 items, each with a 'Select' link, a facility name, and an address. The 'Test Public Facility' entry is highlighted in blue.

Select	South Bend Fire Dept. Station #8	2402 Twykenham Dr.
Select	South Bend Fire Dept. Station #9	2520 Mishawaka Ave.
Select	South Bend Police Department	101 W SAMPLE
Select	St. Joseph Co. Public Library Centre Township Branch	1150 Kern Rd
Select	St. Joseph Co. Public Library Francis Branch	52655 Ironwood Dr.
Select	St. Joseph Co. Public Library German Township Branch	52807 Lynnewood Av
Select	St. Joseph Co. Public Library Lakeville Branch	120 N Michigan St
Select	St. Joseph Co. Public Library LaSalle Branch	3232 Ardmore Tr.
Select	St. Joseph Co. Public Library Main Branch	304 S. Main St.
Select	St. Joseph Co. Public Library North Liberty Branch	105 E Market St
Select	St. Joseph Co. Public Library River Park Branch	2022 Mishawaka Av.
Select	St. Joseph Co. Public Library Tutt Branch	2223 Miami St.
Select	St. Joseph Co. Public Library Western Branch	611 Lombardy Dr
Select	St. Joseph County Jail	701 W SAMPLE
Select	<b>Test Public Facility</b>	<b>Test</b>
Select	Warren Twp. Sta. 1	54837 Quince Road
Select	Warren Twp. Sta. 2	59330 Crumstown Hwy.

Observing that the text boxes are filled with the information for the test public facility

The screenshot shows a web browser window at <http://localhost:59167/Modify.aspx>. The page title is 'MODIFY ITEM'. At the top, there is a navigation bar with links: [Main](#), [Add Item](#), [Modify Item](#), [Delete Item](#), [Display](#), [Search](#), and [Statistics](#). Below the navigation bar, there is a dropdown menu labeled 'Public Facility' and two buttons: 'Back' and 'Save Changes'. The main content area is titled 'Basic Information' and contains a form with the following fields and values:

Basic Information	
Public Facility Name:	Test Public Facility
Type of Public Facility:	Test
Street Address:	Test
City:	Test
State:	Test
ZIP Code:	Test
Latitude:	0.0000000000000000
Longitude:	0.0000000000000000
Phone Number:	Test

Showing the modification to be done to the test public facility

The screenshot shows the same web browser window at <http://localhost:59167/Modify.aspx>. The page title is 'MODIFY ITEM'. At the top, there is a navigation bar with links: [Main](#), [Add Item](#), [Modify Item](#), [Delete Item](#), [Display](#), [Search](#), and [Statistics](#). Below the navigation bar, there is a dropdown menu labeled 'Public Facility' and two buttons: 'Back' and 'Save Changes'. The main content area is titled 'Basic Information' and contains a form with the following fields and values:

Basic Information	
Public Facility Name:	Test Public Facility
Type of Public Facility:	Test
Street Address:	MOAR Test
City:	Test
State:	Test
ZIP Code:	Test
Latitude:	0.0000000000000000
Longitude:	0.0000000000000000
Phone Number:	Test

Showing the completed modification to the test public facility

Select	South Bend Fire Dept. Station #8	2402 Twykenham Dr.
Select	South Bend Fire Dept. Station #9	2520 Mishawaka Ave.
Select	South Bend Police Department	101 W SAMPLE
Select	St. Joseph Co. Public Library Centre Township Branch	1150 Kern Rd
Select	St. Joseph Co. Public Library Francis Branch	52655 Ironwood Dr.
Select	St. Joseph Co. Public Library German Township Branch	52807 Lynnewood Av
Select	St. Joseph Co. Public Library Lakeville Branch	120 N Michigan St
Select	St. Joseph Co. Public Library LaSalle Branch	3232 Ardmore Tr.
Select	St. Joseph Co. Public Library Main Branch	304 S. Main St.
Select	St. Joseph Co. Public Library North Liberty Branch	105 E Market St
Select	St. Joseph Co. Public Library River Park Branch	2022 Mishawaka Av.
Select	St. Joseph Co. Public Library Tutt Branch	2223 Miami St.
Select	St. Joseph Co. Public Library Western Branch	611 Lombardy Dr
Select	St. Joseph County Jail	701 W SAMPLE
Select	Test Public Facility	MOAR Test
Select	Warren Twp. Sta. 1	54837 Quince Road
Select	Warren Twp. Sta. 2	59330 Crumstown Hwy.

Selecting the test business for deletion

DELETE ITEM | [Main](#) | [Add Item](#) | [Modify Item](#) | [Delete Item](#) | [Display](#) | [Search](#) | [Statistics](#)

Business

Select	License Number	Business Name	Street Address
Select	2001-1	Further Test Business	Test
Select	2006-727	LOGAN'S ROADHOUSE	1320 E IRELAND RD
Select	2006-735	HOWARD PARK GENERAL STORE	609 E JEFFERSON BLVD
Select	2006-752	JULIO'S PIZZA	2509 S MICHIGAN ST
Select	2007-1017	PIZZA HUT - DONT USE	2017 SOUTH BEND AVE
Select	2007-1420	INN ON HISTORIC W WASHINGTON	322 W WASHINGTON ST
Select	2008-2578	MTAMBO MBARAKA K	2309 LINCOLN WAY W
Select	2008-2660	JALISCO SUPERMARKET	454 S CARLISLE ST
Select	2008-2822	TONI BERRY	434 NAPOLEON ST
Select	2008-705	CHICORY CAFE CO. - DO NOT USE	105 E JEFFERSON BLVD
Select	2009-1331	CHINA GARDEN RESTAURANT INC	900 E IRELAND RD
Select	2009-3065	UNITED CAB	2730 MISHAWAKA AVE
Select	2009-3137	ABC CAB & COURIER EXPRESS CAB	1733 S MICHIGAN ST

Showing that the test business has been deleted

DELETE ITEM | [Main](#) | [Add Item](#) | [Modify Item](#) | [Delete Item](#) | [Display](#) | [Search](#) | [Statistics](#)

Business

Row successfully deleted.

	<u>License Number</u>	<u>Business Name</u>	<u>Street Address</u>
<a href="#">Select</a>	2006-727	LOGAN'S ROADHOUSE	1320 E IRELAND RD
<a href="#">Select</a>	2006-735	HOWARD PARK GENERAL STORE	609 E JEFFERSON BLVD
<a href="#">Select</a>	2006-752	JULIO'S PIZZA	2509 S MICHIGAN ST
<a href="#">Select</a>	2007-1017	PIZZA HUT - DONT USE	2017 SOUTH BEND AVE
<a href="#">Select</a>	2007-1420	INN ON HISTORIC W WASHINGTON	322 W WASHINGTON ST
<a href="#">Select</a>	2008-2578	MTAMBO MBARAKA K	2309 LINCOLN WAY W
<a href="#">Select</a>	2008-2660	JALISCO SUPERMARKET	454 S CARLISLE ST
<a href="#">Select</a>	2008-2822	TONI BERRY	434 NAPOLEON ST
<a href="#">Select</a>	2008-705	CHICORY CAFE CO. - DO NOT USE	105 E JEFFERSON BLVD
<a href="#">Select</a>	2009-1331	CHINA GARDEN RESTAURANT INC	900 E IRELAND RD
<a href="#">Select</a>	2009-3065	UNITED CAB	2730 MISHAWAKA AVE
<a href="#">Select</a>	2009-3137	ABC CAB & COURIER EXPRESS CAB	1733 S MICHIGAN ST

Selecting the test park for deletion

<a href="#">Select</a>	Randolph Mini Park	1809 Randolph Street
<a href="#">Select</a>	Ravina Park	1522 Lebanon St.
<a href="#">Select</a>	River Manor Park	3811 Glenview Dr.
<a href="#">Select</a>	Riverside Park	3840 Riverside Drive
<a href="#">Select</a>	Rum Village Annex	2700 Gertrude St
<a href="#">Select</a>	Ryer Mini Park	1565 Ryer St.
<a href="#">Select</a>	Seitz Park	340 E. Colfax
<a href="#">Select</a>	Sorin Mini Park	3366 Sorin Street
<a href="#">Select</a>	Southeast Neighborhood Park	503 Wenger Street
<a href="#">Select</a>	Studebaker Golf Course	718 Calvert Street
<a href="#">Select</a>	<b>Test Park</b>	<b>More Test</b>
<a href="#">Select</a>	Vassar Park	1100 Sherman Avenue
<a href="#">Select</a>	Viewing Park	900 Northside Blvd
<a href="#">Select</a>	Voorde Park	3433 Keller St.
<a href="#">Select</a>	Walker Field Park	2212 S. Walnut Street
<a href="#">Select</a>	Westhaven Park	1220 Sussex Drive
<a href="#">Select</a>	Woodlawn Park	2166 Riverside Drive

Showing that the test park has been deleted



<a href="#">Select</a>	Pulaski Park	1308 Huron
<a href="#">Select</a>	Randolph Mini Park	1809 Randolph Street
<a href="#">Select</a>	Ravina Park	1522 Lebanon St.
<a href="#">Select</a>	River Manor Park	3811 Glenview Dr.
<a href="#">Select</a>	Riverside Park	3840 Riverside Drive
<a href="#">Select</a>	Rum Village Annex	2700 Gertrude St
<a href="#">Select</a>	Ryer Mini Park	1565 Ryer St.
<a href="#">Select</a>	Seitz Park	340 E. Colfax
<a href="#">Select</a>	Sorin Mini Park	3366 Sorin Street
<a href="#">Select</a>	Southeast Neighborhood Park	503 Wenger Street
<a href="#">Select</a>	Studebaker Golf Course	718 Calvert Street
<a href="#">Select</a>	Vassar Park	1100 Sherman Avenue
<a href="#">Select</a>	Viewing Park	900 Northside Blvd
<a href="#">Select</a>	Voorde Park	3433 Keller St.
<a href="#">Select</a>	Walker Field Park	2212 S. Walnut Street
<a href="#">Select</a>	Westhaven Park	1220 Sussex Drive
<a href="#">Select</a>	Woodlawn Park	2166 Riverside Drive

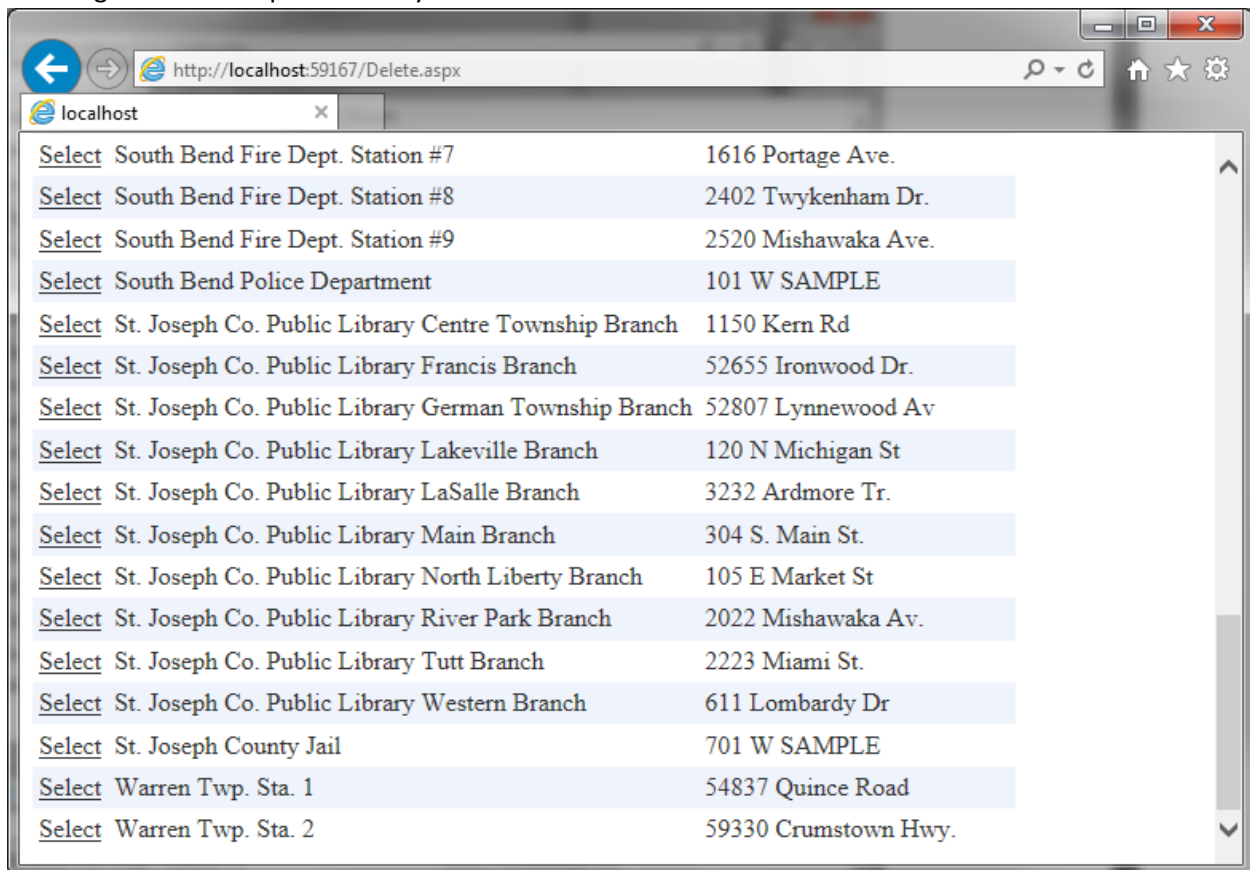
Selecting the test public facility for deletion



<a href="#">Select</a>	South Bend Fire Dept. Station #8	2402 Twykenham Dr.
<a href="#">Select</a>	South Bend Fire Dept. Station #9	2520 Mishawaka Ave.
<a href="#">Select</a>	South Bend Police Department	101 W SAMPLE
<a href="#">Select</a>	St. Joseph Co. Public Library Centre Township Branch	1150 Kern Rd
<a href="#">Select</a>	St. Joseph Co. Public Library Francis Branch	52655 Ironwood Dr.
<a href="#">Select</a>	St. Joseph Co. Public Library German Township Branch	52807 Lynnewood Av
<a href="#">Select</a>	St. Joseph Co. Public Library Lakeville Branch	120 N Michigan St
<a href="#">Select</a>	St. Joseph Co. Public Library LaSalle Branch	3232 Ardmore Tr.
<a href="#">Select</a>	St. Joseph Co. Public Library Main Branch	304 S. Main St.
<a href="#">Select</a>	St. Joseph Co. Public Library North Liberty Branch	105 E Market St
<a href="#">Select</a>	St. Joseph Co. Public Library River Park Branch	2022 Mishawaka Av.
<a href="#">Select</a>	St. Joseph Co. Public Library Tutt Branch	2223 Miami St.
<a href="#">Select</a>	St. Joseph Co. Public Library Western Branch	611 Lombardy Dr
<a href="#">Select</a>	St. Joseph County Jail	701 W SAMPLE
<a href="#">Select</a>	<b>Test Public Facility</b>	<b>MOAR Test</b>
<a href="#">Select</a>	Warren Twp. Sta. 1	54837 Quince Road
<a href="#">Select</a>	Warren Twp. Sta. 2	59330 Crumstown Hwy.



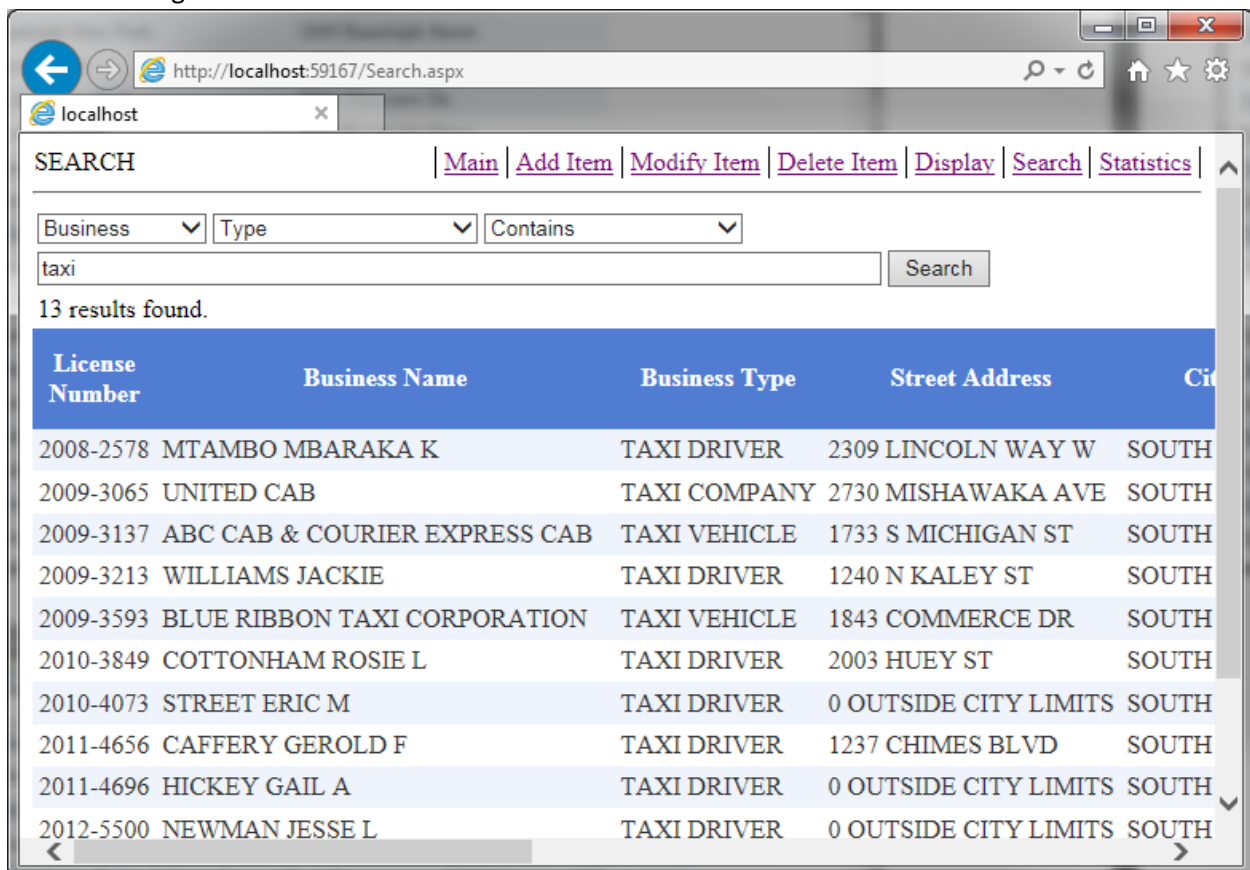
Showing that the test public facility has been deleted



http://localhost:59167/Delete.aspx

Select	South Bend Fire Dept. Station #7	1616 Portage Ave.
Select	South Bend Fire Dept. Station #8	2402 Twykenham Dr.
Select	South Bend Fire Dept. Station #9	2520 Mishawaka Ave.
Select	South Bend Police Department	101 W SAMPLE
Select	St. Joseph Co. Public Library Centre Township Branch	1150 Kern Rd
Select	St. Joseph Co. Public Library Francis Branch	52655 Ironwood Dr.
Select	St. Joseph Co. Public Library German Township Branch	52807 Lynnewood Av
Select	St. Joseph Co. Public Library Lakeville Branch	120 N Michigan St
Select	St. Joseph Co. Public Library LaSalle Branch	3232 Ardmore Tr.
Select	St. Joseph Co. Public Library Main Branch	304 S. Main St.
Select	St. Joseph Co. Public Library North Liberty Branch	105 E Market St
Select	St. Joseph Co. Public Library River Park Branch	2022 Mishawaka Av.
Select	St. Joseph Co. Public Library Tutt Branch	2223 Miami St.
Select	St. Joseph Co. Public Library Western Branch	611 Lombardy Dr
Select	St. Joseph County Jail	701 W SAMPLE
Select	Warren Twp. Sta. 1	54837 Quince Road
Select	Warren Twp. Sta. 2	59330 Crumstown Hwy.

Demonstrating various searches



http://localhost:59167/Search.aspx

SEARCH | [Main](#) | [Add Item](#) | [Modify Item](#) | [Delete Item](#) | [Display](#) | [Search](#) | [Statistics](#)

Business ▼ Type ▼ Contains ▼

taxi Search

13 results found.

License Number	Business Name	Business Type	Street Address	City
2008-2578	MTAMBO MBARAKA K	TAXI DRIVER	2309 LINCOLN WAY W	SOUTH
2009-3065	UNITED CAB	TAXI COMPANY	2730 MISHAWAKA AVE	SOUTH
2009-3137	ABC CAB & COURIER EXPRESS CAB	TAXI VEHICLE	1733 S MICHIGAN ST	SOUTH
2009-3213	WILLIAMS JACKIE	TAXI DRIVER	1240 N KALEY ST	SOUTH
2009-3593	BLUE RIBBON TAXI CORPORATION	TAXI VEHICLE	1843 COMMERCE DR	SOUTH
2010-3849	COTTONHAM ROSIE L	TAXI DRIVER	2003 HUEY ST	SOUTH
2010-4073	STREET ERIC M	TAXI DRIVER	0 OUTSIDE CITY LIMITS	SOUTH
2011-4656	CAFFERY GEROLD F	TAXI DRIVER	1237 CHIMES BLVD	SOUTH
2011-4696	HICKEY GAIL A	TAXI DRIVER	0 OUTSIDE CITY LIMITS	SOUTH
2012-5500	NEWMAN JESSE L	TAXI DRIVER	0 OUTSIDE CITY LIMITS	SOUTH

## Demonstrating various searches

http://localhost:59167/Search.aspx

SEARCH | [Main](#) | [Add Item](#) | [Modify Item](#) | [Delete Item](#) | [Display](#) | [Search](#) | [Statistics](#)

Business Type Does Not Contain

taxi Search

37 results found.

License Number	Business Name	Business Type	Street
2006-727	LOGAN'S ROADHOUSE	RESTAURANTS A-M	1320 E IRELAND
2006-735	HOWARD PARK GENERAL STORE	RESTAURANTS A-M	609 E JEFFERSON
2006-752	JULIO'S PIZZA	RESTAURANTS A-M	2509 S MICHIGAN
2007-1017	PIZZA HUT - DONT USE	RESTAURANTS N-Z	2017 SOUTH BEND
2007-1420	INN ON HISTORIC W WASHINGTON	RESTAURANTS A-M	322 W WASHINGTON
2008-2660	JALISCO SUPERMARKET	RESTAURANTS A-M	454 S CARLISLE
2008-2822	TONI BERRY	LAWN PARK UNDER 10 SPACES	434 NAPOLEON
2008-705	CHICORY CAFE CO. - DO NOT USE	RESTAURANTS A-M	105 E JEFFERSON
2009-1331	CHINA GARDEN RESTAURANT INC	RESTAURANTS A-M	900 E IRELAND
2009-3666	1105 N ST. LOUIS BLVD	LAWN PARK UNDER 10 SPACES	1105 N SAINT LOUIS

## Demonstrating various searches

http://localhost:59167/Search.aspx

SEARCH | [Main](#) | [Add Item](#) | [Modify Item](#) | [Delete Item](#) | [Display](#) | [Search](#) | [Statistics](#)

Park # of Baseball Diamonds Is Not Equal To

0 Search

7 results found.

Park Name	Park Type	Street Address	City	State	ZIP Code	Latitude
Belleville Park	Community Park	1300 S. Mayflower Road	South Bend	Indiana	46619	41.660032260000490
Boehm Park	Community Park	3205 Edison Road	South Bend	Indiana	46617	41.694669059000034
Booth Tarkington	Neighborhood Park	1722 Rerick Drive	South Bend	Indiana	46635	41.698712303000490
Kennedy Park	Community Park	2700 Westmoor Street	South Bend	Indiana	46628	41.684674648000450
Marshall Park	Neighborhood Park	3725 Springbrook	South Bend	Indiana	46614	41.636660753000060
Potawatomi Park	Community Park	2000 Wall Street	South Bend	Indiana	46615	41.667536510000446
Walker Field Park	Neighborhood Park	2212 S. Walnut Street	South Bend	Indiana	46613	41.651088682000080

## Demonstrating various searches

SEARCH | [Main](#) | [Add Item](#) | [Modify Item](#) | [Delete Item](#) | [Display](#) | [Search](#) | [Statistics](#)

Park # of Golf Courses Is Equal To

0.5 Search

1 result found.

Park Name	Park Type	Street Address	City	State	ZIP Code	Latitude	Longitude
Studebaker Golf Course	Golf Course	718 Calvert Street	South Bend	Indiana	46613	41.654116630000490	-86.259

## Demonstrating various searches

SEARCH | [Main](#) | [Add Item](#) | [Modify Item](#) | [Delete Item](#) | [Display](#) | [Search](#) | [Statistics](#)

Public Facility City Is Equal To

mishawaka Search

11 results found.

Name	Type	StreetAddress	City	State
Mishawaka EMS Station	FIRE STATION	333 E. Mishawaka Ave	Mishawaka	Indiana
Mishawaka Fire Dept. Station #1	FIRE STATION	600 Union St.	Mishawaka	Indiana
Mishawaka Fire Dept. Station #2	FIRE STATION	2332 N. Main St.	Mishawaka	Indiana
Mishawaka Fire Dept. Station #3	FIRE STATION	333 E Douglas Road	Mishawaka	Indiana
Mishawaka Fire Dept. Station #4	FIRE STATION	3000 Harrison Rd	Mishawaka	Indiana
Mishawaka Police Department	POLICE STATION	200 N CHURCH ST	Mishawaka	Indiana
Mishawaka Public Library	LIBRARY	209 Lincoln Way East	Mishawaka	Indiana
Mishawaka Public Library Bittersweet Branch	LIBRARY	602 Bittersweet Rd.	Mishawaka	Indiana
Penn North Fire	FIRE STATION	13750 McKinley	Mishawaka	Indiana
Penn South Sta. 1	FIRE STATION	14940 Jackson Road	Mishawaka	Indiana
Penn South Sta. 2	FIRE STATION	10701 Ireland Road	Mishawaka	Indiana



## Demonstrating various searches

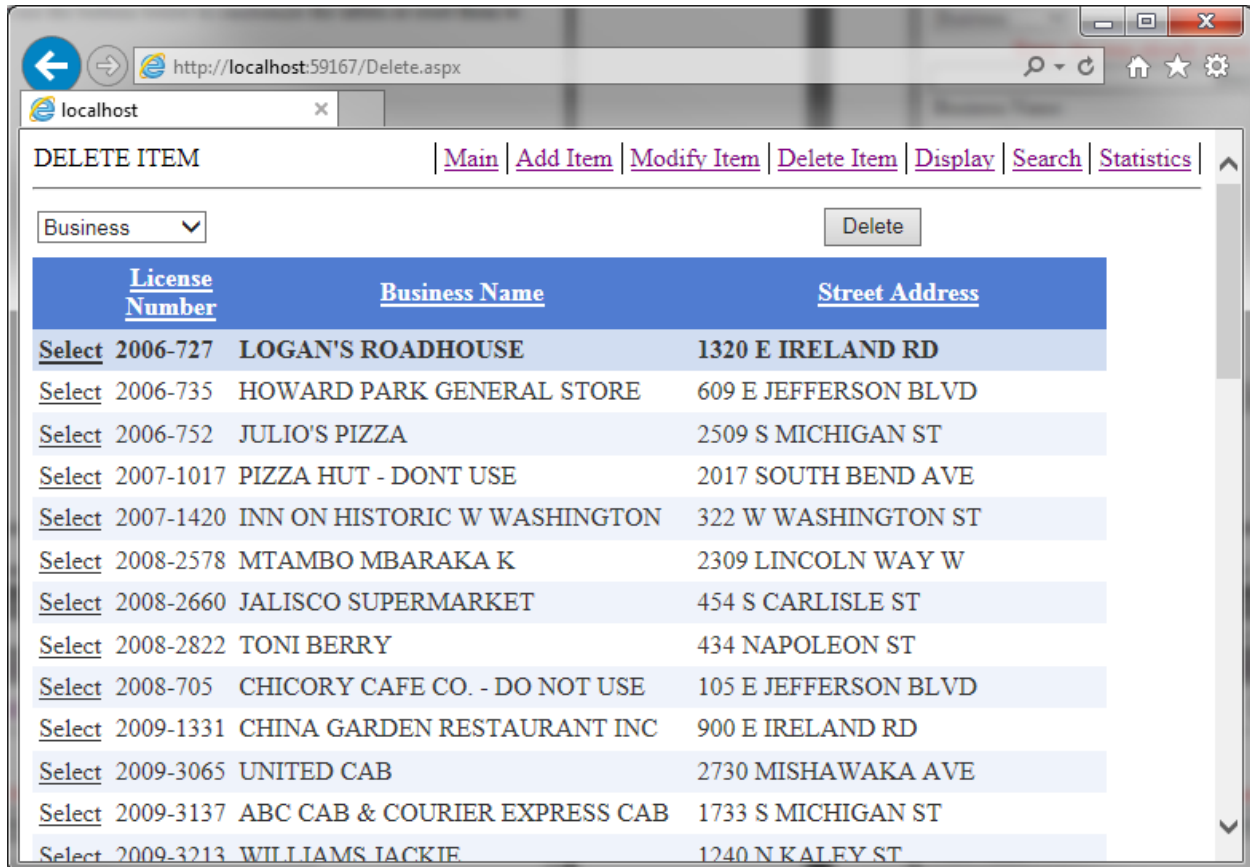
The screenshot shows a web browser window with the address bar displaying `http://localhost:59167/Search.aspx`. The page has a navigation bar with links: [Main](#), [Add Item](#), [Modify Item](#), [Delete Item](#), [Display](#), [Search](#), and [Statistics](#). Below the navigation bar is a search form with three dropdown menus: "Public Facility" (selected), "Type" (selected), and "Contains" (selected). A text input field contains the word "police", and a "Search" button is to its right. Below the search form, it says "3 results found." and displays a table with the following data:

Name	Type	StreetAddress	City	State	Zip	Latitude
Mishawaka Police Department	POLICE STATION	200 N CHURCH ST	Mishawaka	Indiana	46544	41.66218651
South Bend Police Department	POLICE STATION	101 W SAMPLE	South Bend	Indiana	46619	41.66452382
St. Joseph County Jail	POLICE STATION	701 W SAMPLE	South Bend	Indiana	46625	41.67905056

## Showing the statistics

<a href="#">Main</a>   <a href="#">Add Item</a>   <a href="#">Modify Item</a>   <a href="#">Delete Item</a>   <a href="#">Display</a>   <a href="#">Search</a>   <a href="#">Statistics</a>	
Total # of Parks:	50
Total # of Parks by Park Types:	Block Park: 15 Cemetery: 2 Community Park: 8 Golf Course: 4 Memorial: 1 Neighborhood Park: 19 Zoo: 1
Total # of Businesses:	50
Total # of license renewals for each Business:	110 E Angela Blvd: 1 Alarms By Turner: 1 Arborcare Inc: 1 Burton's Resale: 1 Chicory Cafe Co. - Do Not Use: 1 China Garden Restaurant Inc: 1 El Paraiso: 1 Golden Dragon Chinese Rest: 1 Howard Park General Store: 1 Howard Park Ice Rink: 1 Inn On Historic W Washington: 1 Jalisco Supermarket: 1 Jp Morgan Chase Corporate Sec: 1 Julio's Pizza: 1 Let's Spoon Frozen Yogurt / CI: 1 Lil J Bueno's Body Shop & Auto: 1 Logan's Roadhouse: 1 Main Street Coffee House Inc: 1 Meijer Gas Station #120: 1 Mibu Servicer Inc: 1 Pizza Hut - Dont Use: 1 Residence Inn: 1 St Joseph Co 4-H Fair Inc: 1 That Cheesecake Place: 1 Tom's Car Care Center Inc: 1 Torch Lounge Inc: 1
Total # of Facilities that have substring "Fire":	22

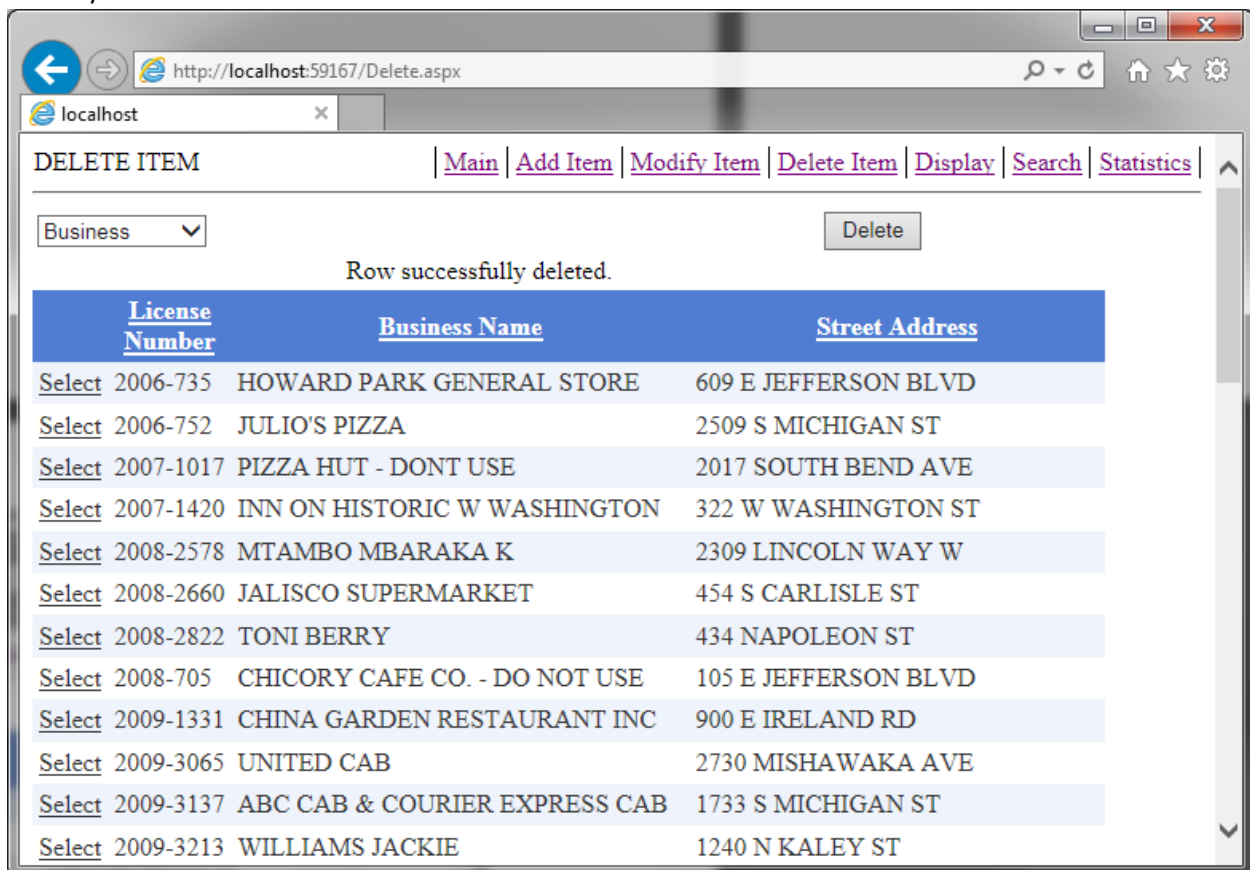
Selecting a sacrificial business for deletion to test the reset functionality



The screenshot shows a web browser window with the URL <http://localhost:59167/Delete.aspx>. The page has a navigation bar with links: [Main](#), [Add Item](#), [Modify Item](#), [Delete Item](#), [Display](#), [Search](#), and [Statistics](#). Below the navigation bar, there is a dropdown menu labeled "Business" and a "Delete" button. The main content area displays a table with three columns: "License Number", "Business Name", and "Street Address". Each row in the table has a "Select" link to its left. The table contains 13 rows of business data.

	<u>License Number</u>	<u>Business Name</u>	<u>Street Address</u>
<a href="#">Select</a>	2006-727	LOGAN'S ROADHOUSE	1320 E IRELAND RD
<a href="#">Select</a>	2006-735	HOWARD PARK GENERAL STORE	609 E JEFFERSON BLVD
<a href="#">Select</a>	2006-752	JULIO'S PIZZA	2509 S MICHIGAN ST
<a href="#">Select</a>	2007-1017	PIZZA HUT - DONT USE	2017 SOUTH BEND AVE
<a href="#">Select</a>	2007-1420	INN ON HISTORIC W WASHINGTON	322 W WASHINGTON ST
<a href="#">Select</a>	2008-2578	MTAMBO MBARAKA K	2309 LINCOLN WAY W
<a href="#">Select</a>	2008-2660	JALISCO SUPERMARKET	454 S CARLISLE ST
<a href="#">Select</a>	2008-2822	TONI BERRY	434 NAPOLEON ST
<a href="#">Select</a>	2008-705	CHICORY CAFE CO. - DO NOT USE	105 E JEFFERSON BLVD
<a href="#">Select</a>	2009-1331	CHINA GARDEN RESTAURANT INC	900 E IRELAND RD
<a href="#">Select</a>	2009-3065	UNITED CAB	2730 MISHAWAKA AVE
<a href="#">Select</a>	2009-3137	ABC CAB & COURIER EXPRESS CAB	1733 S MICHIGAN ST
<a href="#">Select</a>	2009-3213	WILLIAMS JACKIE	1240 N KALEY ST

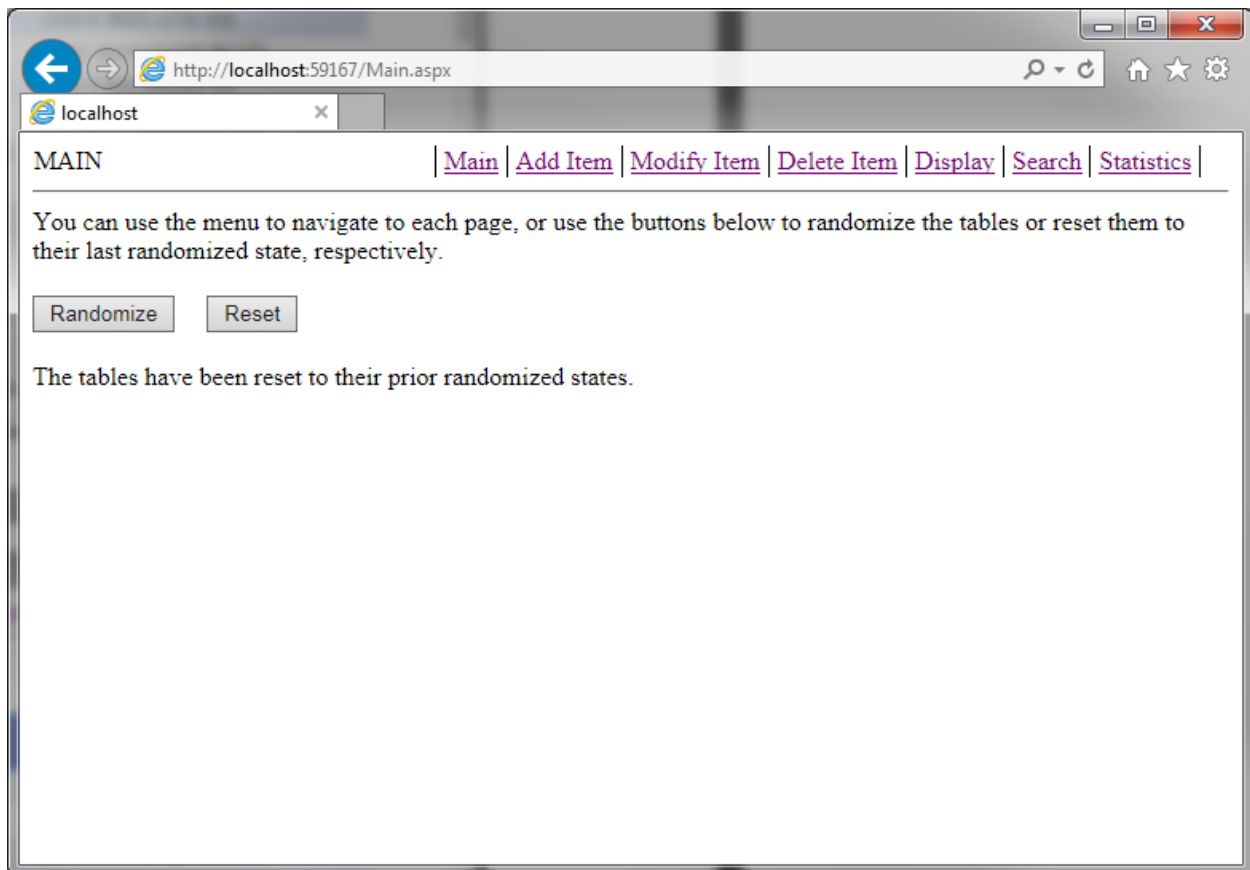
Goodbye



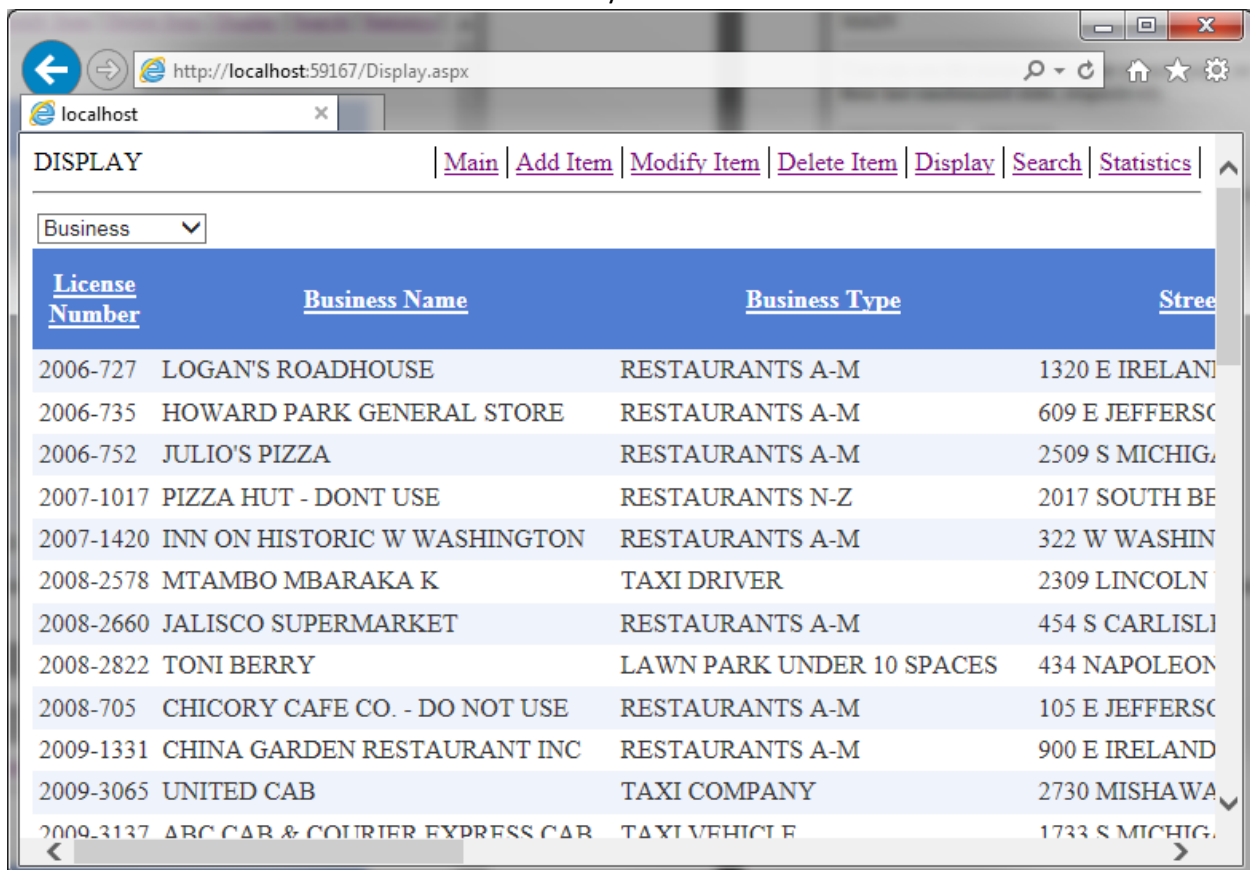
The screenshot shows the same web browser window after a business has been deleted. The message "Row successfully deleted." is displayed above the table. The table now contains 12 rows of business data, as the first row from the previous screenshot has been removed.

	<u>License Number</u>	<u>Business Name</u>	<u>Street Address</u>
<a href="#">Select</a>	2006-735	HOWARD PARK GENERAL STORE	609 E JEFFERSON BLVD
<a href="#">Select</a>	2006-752	JULIO'S PIZZA	2509 S MICHIGAN ST
<a href="#">Select</a>	2007-1017	PIZZA HUT - DONT USE	2017 SOUTH BEND AVE
<a href="#">Select</a>	2007-1420	INN ON HISTORIC W WASHINGTON	322 W WASHINGTON ST
<a href="#">Select</a>	2008-2578	MTAMBO MBARAKA K	2309 LINCOLN WAY W
<a href="#">Select</a>	2008-2660	JALISCO SUPERMARKET	454 S CARLISLE ST
<a href="#">Select</a>	2008-2822	TONI BERRY	434 NAPOLEON ST
<a href="#">Select</a>	2008-705	CHICORY CAFE CO. - DO NOT USE	105 E JEFFERSON BLVD
<a href="#">Select</a>	2009-1331	CHINA GARDEN RESTAURANT INC	900 E IRELAND RD
<a href="#">Select</a>	2009-3065	UNITED CAB	2730 MISHAWAKA AVE
<a href="#">Select</a>	2009-3137	ABC CAB & COURIER EXPRESS CAB	1733 S MICHIGAN ST
<a href="#">Select</a>	2009-3213	WILLIAMS JACKIE	1240 N KALEY ST

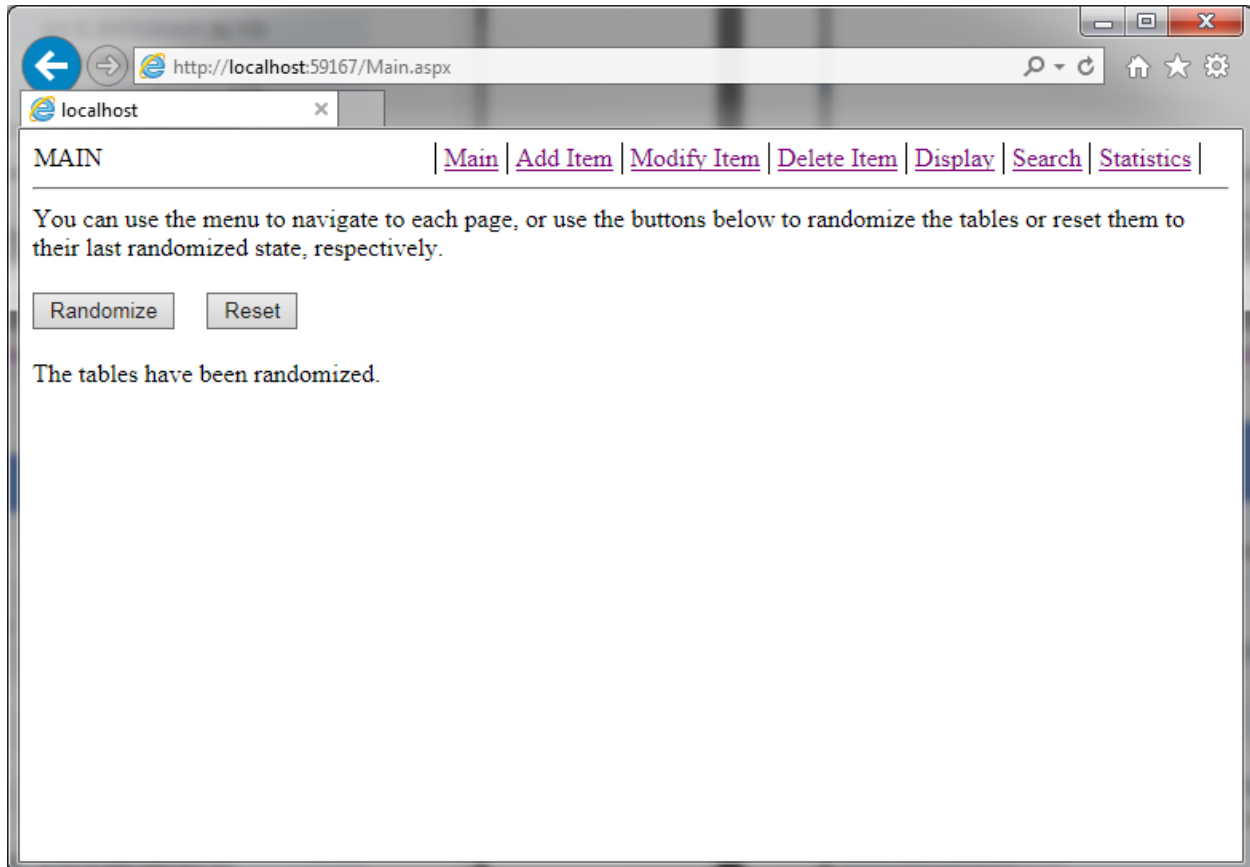
Tables have been reset



Sacrificial business has made a miraculous recovery



## Randomize everything



## Presto-chango, different data

