```
/*-----------------------------------------------------------------------------------------------
 * Author:      Dan Cassidy
 * Date:        2015-06-23
 * Assignment:  cView-P4
 * Source File: Add.aspx.cs
 * Language:    C#
 * Course:      CSCI-C 490, C# Programming, MoWe 08:00
 * Purpose:     Code-behind file for Add.aspx. Controls the process of adding an item to the
 *              database via the Entity Framework model.
-----------------------------------------------------------------------------------------------*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace cView_P4_DanCassidy
{
    public partial class Add : System.Web.UI.Page
    {
        /*---------------------------------------------------------------------------------------
         * Name:    btnAdd_Click
         * Type:    Event Handler Method
         * Purpose: Handles constructing, errorchecking, and finally adding an object to the
         *          database.
         * Input:   object sender, holds a reference to the object that raised this event.
         * Input:   EventArgs e, holds data related to this event.
         * Output:  Nothing.
        ---------------------------------------------------------------------------------------*/
        protected void btnAdd_Click(object sender, EventArgs e)
        {
            object toAdd = null;

            switch ((Global.Enums.ItemTypes)ddlItemType.SelectedIndex)
            {
                case Global.Enums.ItemTypes.Business:
                    toAdd = new Business()
                    {
                        // Common Fields
                        Name = txtName.Text.Trim(),
                        Type = txtType.Text.Trim(),
                        StreetAddress = txtStreetAddress.Text.Trim(),
                        City = txtCity.Text.Trim(),
                        State = txtState.Text.Trim(),
                        Zip = txtZip.Text.Trim(),
                        Latitude = SimpleConvert.ToDecimal(txtLatitude.Text.Trim()),
                        Longitude = SimpleConvert.ToDecimal(txtLongitude.Text.Trim()),
                        Phone = txtPhone.Text.Trim(),

                        // Business Fields
                        LicenseNumber = txtLicenseNumber.Text.Trim(),
                        LicenseIssueDate = SimpleConvert.ToDateTime(
                            txtLicenseExpirDate.Text.Trim()),
                        LicenseExpirDate = SimpleConvert.ToDateTime(
                            txtLicenseExpirDate.Text.Trim()),
                        LicenseStatus = txtLicenseStatus.Text.Trim(),
                        CouncilDistrict = txtCouncilDistrict.Text.Trim()
                    };
                    break;

                case Global.Enums.ItemTypes.Park:
                    toAdd = new Park()
                    {
                        // Common Fields
                        Name = txtName.Text.Trim(),
                        Type = txtType.Text.Trim(),
                        StreetAddress = txtStreetAddress.Text.Trim(),
                        City = txtCity.Text.Trim(),
                        State = txtState.Text.Trim(),
                        Zip = txtZip.Text.Trim(),
                        Latitude = SimpleConvert.ToDecimal(txtLatitude.Text.Trim()),
                        Longitude = SimpleConvert.ToDecimal(txtLongitude.Text.Trim()),
                        Phone = txtPhone.Text.Trim(),

                        // Park Fields
                        FeatureBaseball = SimpleConvert.ToByte(txtFeatureBaseball.Text.Trim()),
                        FeatureBasketball = SimpleConvert.ToDecimal(
                            txtFeatureBasketball.Text.Trim()),
                        FeatureGolf = SimpleConvert.ToDecimal(txtFeatureGolf.Text.Trim()),
                        FeatureLargeMPField = SimpleConvert.ToByte(
                            txtFeatureLargeMPField.Text.Trim()),
                        FeatureTennis = SimpleConvert.ToByte(txtFeatureTennis.Text.Trim()),
                        FeatureVolleyball = SimpleConvert.ToByte(txtFeatureVolleyball.Text.Trim())
                    };
                    break;

                case Global.Enums.ItemTypes.PublicFacility:
                    toAdd = new PublicFacility()
                    {
                        // Common Fields
                        Name = txtName.Text.Trim(),
                        Type = txtType.Text.Trim(),
                        StreetAddress = txtStreetAddress.Text.Trim(),
                        City = txtCity.Text.Trim(),
                        State = txtState.Text.Trim(),
                        Zip = txtZip.Text.Trim(),
                        Latitude = SimpleConvert.ToDecimal(txtLatitude.Text.Trim()),
                        Longitude = SimpleConvert.ToDecimal(txtLongitude.Text.Trim()),
                        Phone = txtPhone.Text.Trim()
                    };
                    break;

                default:
                    // ... How?
                    return;
            }

            // Add the object to the database.
            if (toAdd != null)
            {
                try
                {
                    using (CViewDataEntities database = new CViewDataEntities())
                    {
                        if (toAdd is Business)
                        {
                            Business businessToAdd = toAdd as Business;

                            // Do error-checking.
                            if (businessToAdd.LicenseNumber == string.Empty ||
                                businessToAdd.LicenseNumber == null)
                            {
                                throw new Global.Exceptions.EmptyOrNullPKException(
                                    Global.Strings.BusinessKey);
                            }
                            else if (database.Businesses.Find(businessToAdd.LicenseNumber) != null)
                            {
                                throw new Global.Exceptions.DuplicatePKException(
                                    Global.Strings.BusinessKey, businessToAdd.LicenseNumber);
                            }

                            // If everything is ok, add item to table.
                            database.Businesses.Add(businessToAdd);
                        }
                        else if (toAdd is Park)
                        {
                            Park parkToAdd = toAdd as Park;

                            //Do error-checking.
                            if (parkToAdd.Name == string.Empty ||
                                parkToAdd.Name == null)
                            {
                                throw new Global.Exceptions.EmptyOrNullPKException(
                                    Global.Strings.ParkKey);
                            }
                            else if (database.Parks.Find(parkToAdd.Name) != null)
                            {
                                throw new Global.Exceptions.DuplicatePKException(
                                    Global.Strings.ParkKey, parkToAdd.Name);
                            }

                            // If everything is ok, add item to table.
                            database.Parks.Add(parkToAdd);
                        }
                        else if (toAdd is PublicFacility)
                        {
                            PublicFacility publicFacilityToAdd = toAdd as PublicFacility;

                            //Do error-checking.
                            if (publicFacilityToAdd.Name == string.Empty ||
                                publicFacilityToAdd.Name == null)
                            {
                                throw new Global.Exceptions.EmptyOrNullPKException(
                                    Global.Strings.PublicFacilityKey);
                            }
                            else if (database.PublicFacilities.Find(publicFacilityToAdd.Name) !=
                                     null)
                            {
                                throw new Global.Exceptions.DuplicatePKException(
                                    Global.Strings.PublicFacilityKey, publicFacilityToAdd.Name);
                            }

                            // If everything is ok, add item to table.
                            database.PublicFacilities.Add(publicFacilityToAdd);
                        }

                        database.SaveChanges();

                        lblResult.Text = "Added record to the table.";
                        lblResult.Visible = true;
                        lblError.Visible = false;
                    }
                }
                catch (Exception ex)
                {
                    // Drill down to the innermost exception.
                    while (ex.InnerException != null)
                        ex = ex.InnerException;

                    lblError.Text = "Error: " + ex.Message;
                    lblError.Visible = true;
                    lblResult.Visible = false;
                }
            }
        }

        /*-------------------------------------------------------------------------------------------
         * Name:    ddlItemType_SelectedIndexChanged
         * Type:    Event Handler Method
         * Purpose: Handles showing and hiding the various controls to allow a user to enter
         *          information so that an item can be added to the database.
         * Input:   object sender, holds a reference to the object that raised this event.
         * Input:   EventArgs e, holds data related to this event.
         * Output:  Nothing.
        -------------------------------------------------------------------------------------------*/
        protected void ddlItemType_SelectedIndexChanged(object sender, EventArgs e)
        {
            // Hide things until needed.
            lblError.Visible = false;
            lblResult.Visible = false;

            mViewBasic.ActiveViewIndex = -1;
            mViewSpecific.ActiveViewIndex = -1;
            btnAdd.Visible = false;

            // Decide which controls should be shown to enable user to add an item.
            switch ((Global.Enums.ItemTypes)ddlItemType.SelectedIndex)
            {
                case Global.Enums.ItemTypes.Business:
                    mViewBasic.ActiveViewIndex = 0;
                    mViewSpecific.ActiveViewIndex = 0;
                    lblName.Text = Global.Strings.BusinessName + Global.Strings.Separator;
                    lblType.Text = Global.Strings.BusinessType + Global.Strings.Separator;
                    btnAdd.Visible = true;
                    break;

                case Global.Enums.ItemTypes.Park:
                    mViewBasic.ActiveViewIndex = 0;
                    mViewSpecific.ActiveViewIndex = 1;
                    lblName.Text = Global.Strings.ParkName + Global.Strings.Separator;
                    lblType.Text = Global.Strings.ParkType + Global.Strings.Separator;
                    btnAdd.Visible = true;
                    break;

                case Global.Enums.ItemTypes.PublicFacility:
                    mViewBasic.ActiveViewIndex = 0;
                    mViewSpecific.ActiveViewIndex = -1;
                    lblName.Text = Global.Strings.PublicFacilityName + Global.Strings.Separator;
                    lblType.Text = Global.Strings.PublicFacilityType + Global.Strings.Separator;
                    btnAdd.Visible = true;
                    break;

                default:
                    break;
            }
        }
    }
}
```