**CHƯƠNG 1: CHUẨN BỊ VÀ CÀI ĐẶT**

**Bài 1.1. Cài đặt Visual Studio**

Video:

<https://youtu.be/hto2TVEhNkA>

Video test chương trinh:

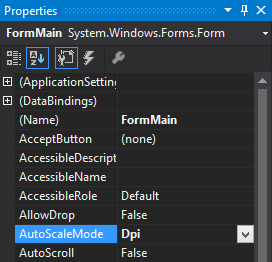
<https://youtu.be/sqfZNeMo_x4>

**CHƯƠNG 2: LẬP TRÌNH WINDOWS FORM**

# Bài 2.1. Tổng quan về windows form

# Chú ý: Giao diện bị mờ trên màn hình có DPI cao. Cách khắc phục:

# ở form chỉnh thuộc tính: AutoScaleMode từ Font sang Dpi



* mở file Program.cs , rồi khai báo trong class Program

|  |  |
| --- | --- |
| 1  2 | [System.Runtime.InteropServices.**DllImport**("user32.dll")]  **private** **static** **extern** **bool** **SetProcessDPIAware**(); |

* Và thêm đoạn code này vào trong main:

**if** (Environment.OSVersion.Version.Major >= 6)

{

**SetProcessDPIAware**();

}

Video:

<https://youtu.be/1qp_zkujZ-4>

#### **Các khái niệm**

* Một form là cửa sổ cha, bề mặt hiển thị dùng để chứa các view, control khác.
* Một control là phần tử giao diện đơn lẻ sử dụng để hiển thị, nhận tương tác từ người dùng như các nút bấm, ô nhập dữ liệu đầu vào.
* Lập trình giao diện windows forms là dùng các điều khiển, form để tạo ra ứng dụng phục vụ mục đích sử dụng nào đó.

#### **Cách tạo project window forms:**

* Từ giao diện Solution Explorer chuột phải vào tên Solution -> Add -> New Project…
* Trên giao diện hiện ra chọn ngôn ngữ là C#. Sau đó chọn Windows Forms App(.Net Framework) -> Enter hoặc nhấn Next.

Text, application

Description automatically generated with medium confidence

* Trên giao diện mới hiện ra. Đặt tên cho project ở mục Project name và nhấn Create.

Graphical user interface, text, application

Description automatically generated

* Thực hiện các thao tác khác trên project vừa tạo.

#### **Các cửa số trong giao diện windows forms**

* Cửa số Toolbox chứa các thành phần giao diện gom thành từng nhóm. Click vào tam giác ở đầu dòng để xem nội dung chi tiết.
* Nếu Toolbox không hiển thị. Bạn vào View -> Toolbox trên thanh công cụ.
* Cửa sổ Solution Explorer chứa danh sách các project, file, thư mục của từng project.
* Cửa sổ Properties chứa các đặc trưng, thuộc tính của form, controls. Đây là nơi ta thực hiện các thay đổi giá trị thuộc tính cho form hoặc control.

# Bài 2.2. Các thao tác cơ bản trên forms

Video:

<https://youtu.be/mAt_QzJzUHo>

#### **Các thuộc tính của form**

* **BackColor**: màu background.
* **BackgroundImage**: ảnh dùng làm background của form.
* **BackgroundImageLayout**: thiết lập chế độ co dãn ảnh cho khớp với kích thước của form.
* **Cursor**: cách thức hiển thị con trỏ chuột.
* **Font**: cho phép thay đổi font, kiểu chữ, kích thước chữ.
* **ForeColor**: màu chữ.
* **Text**: Tiêu đề của form.
* **UserWaitCursor**: giao diện con trỏ ở chế độ chờ.
* **Enable**: cho phép tương tác với form và các control trên form.
* **Name**: tên form, thường là tên lớp.
* **AutoScaleMode**: tự động co dãn.
* **AutoSize**: cho phép tự động co dãn kích thước form.
* **AutoSizeMode**: chọn cách thức co dãn kích thước form.
* **Location**: vị trí của form.
* **MaximumSize**: kích thước form tối đa theo 2 chiều.
* **MinimumSize**: kích thước form tối thiểu theo 2 chiều.
* **Padding**: lề tính từ viền vào tới nội dung bên trong.
* **Size**: kích thước form theo chiều dài và rộng.
* **StartPosition**: vị trí xuất hiện của form.
* **WindowState**: trạng thái hiện thời của form, có thể bình thường hoặc thu nhỏ, phóng to.
* **ControlBox**: cho phép hiển thị nút điều khiển form(thu phóng, tắt).
* **HelpButton**: cho phép hiển thị nút trợ giúp.
* **Icon**: biểu tượng của form.
* **MaximizeBox**: hiển thị nút hộp thoại phóng to.
* **MinimizeBox**: hiển thị hộp thoại thu nhỏ.
* **Opacity**: độ trong suốt của form.
* **ShowTaskbar**: hiển thị biểu tượng ứng dụng trên thanh  taskbar nếu ứng dụng đang chạy.
* **TopMost**: cho phép hiển thị giao diện form trên cùng so với các giao diện form khác.

# Bài 2.3. Thêm mới và thiết lập form để khởi chạy

Video:

<https://youtu.be/TpxkHquiZLs>

# Bài 2.4. Tìm hiểu control Button

Video:

<https://youtu.be/yfEn3WvMICQ>

# Bài 2.5. Tìm hiểu control Label và TextBox

Video:

<https://youtu.be/sd2MVvpwuG8>

#### **Mục đích sử dụng**

* Sử dụng label để hiển thị các đoạn văn bản, hình ảnh chỉ đọc, chỉ để xem.
* Sử dụng textbox để cho phép nhập dữ liệu dạng kí tự, văn bản từ bên ngoài vào.
* Sử dụng thuộc tính Text của label/textbox để gán/lấy giá trị văn bản hiển thị trên control tương ứng.
* Sử dụng thuộc tính Enabled để cho phép tương tác/không tương tác với textbox.

# Bài 2.6. CheckBox, RadioButton và GroupBox

Video:

<https://youtu.be/Mr2tE3lf1jo>

#### **Mục đích sử dụng:**

* Sử dụng checkbox để cho phép chọn hoặc bỏ chọn 0, 1 hoặc nhiều lựa chọn. Ví dụ chọn các món ăn yêu thích để gọi món.
* Sử dụng radio button để chọn 1 trong các tiêu chí/lựa chọn. Ví dụ: chọn giới tính.
* Sử dụng groupbox để gom nhóm các button lựa chọn lại thành từng nhóm có liên quan đến nhau và đặt cho chúng 1 cái nhãn chung.
* Sử dụng thuộc tính Checked để xét xem checkbox hay radio button có được chọn không.
* Sử dụng sự kiện click hoặc checkchanged để bắt sự kiện cho checkbox hoặc radio button.

# Bài 2.7. ComboBox

Video:

<https://youtu.be/7uyPCOPL4kM>

#### **Những điểm cần lưu ý**

* ComboBox là sự kết hợp giữa TextBox và ListBox. Trong đó cho phép ta chọn và sửa 1 phần tử cụ thể nào đó.
* Ta có thể thêm, chèn, xóa phần tử hoặc xóa toàn bộ các phần tử khỏi combobox.
* Để lấy phần tử đã chọn từ combobox ta gọi thuộc tính SelectedItem.
* Để lấy/gán chỉ số phần tử nào đó trong ComboBox ta gọi thuộc tính SelectedIndex.
* Để lấy/gán dữ liệu trên ô TextBox của ComboBox ta gọi thuộc tính Text.
* Để thiết lập danh sách các phần tử hiển thị trên combobox, ta có thể edit trực tiếp trong mục Items của hộp thoại Properties.
* Để thêm các phần tử vào tập dữ liệu của combobox ta có thể sử dụng Items và các phương thức như Add, AddRange, Insert.
* Hoặc thiết lập tập các phần tử qua DataSource. Đoạn code sau minh họa việc sử dụng DataSource.

public partial class LearnComboBox : Form

{

  public List<string> Brands { get; set; } = new List<string>(

      new string[] { "Honda", "Toyota", "Nissan", "Mazda",

          "Huyndai", "Audi", "Subaru"}

      );

  public LearnComboBox()

  {

      InitializeComponent();

      comboBoxBrand.DataSource = Brands;

      comboBoxBrand.SelectedIndex = -1;

  }

  private void button1\_Click(object sender, EventArgs e)

  {

      MessageBox.Show("Bạn chọn hãng: " + comboBoxBrand.Text);

  }

}

* Đoạn code sau minh họa việc thiết lập trực tiếp dữ liệu cho Items qua phương thức Add, AddRange:

public partial class LearnComboBox : Form

{

  public List<string> Brands { get; set; } = new List<string>(

      new string[] { "Honda", "Toyota", "Nissan", "Mazda",

          "Huyndai", "Audi", "Subaru"}

      );

    public LearnComboBox()

  {

      InitializeComponent();

      //comboBoxBrand.DataSource = Brands;

      comboBoxBrand.SelectedIndex = -1;

**comboBoxBrand.Items.AddRange(Brands.ToArray()); // thêm mảng các phần tử vào tập phần tử**

  }

    private void button1\_Click(object sender, EventArgs e)

  {

      MessageBox.Show("Bạn chọn hãng: " + comboBoxBrand.Text);

  }

    private void button2\_Click(object sender, EventArgs e)

  {

**comboBoxBrand.Items.Add(textBoxBrand.Text); // thêm 1 phần tử mới vào cuối tập phần tử ComboBox**

  }

}

# Bài 2.8. ListBox

Video:

<https://youtu.be/egZ16KrpfqA>

#### **Tìm hiểu về ListBox**

* ListBox cho phép hiển thị và lựa chọn 1 hoặc nhiều phần tử từ danh sách cho trước.
* Danh sách này không cho phép sửa giá trị các phần tử sau khi chọn như ComboBox.
* Để thay đổi chế độ cho phép lựa chọn phần tử, bạn thay đổi giá trị của thuộc tính SelectionMode.
* Để thiết lập giá trị danh sách các phần tử hiển thị trong SelectionMode, ta sử dụng thuộc tính Items, DataSource.
* Khi gán giá trị cho DataSource ta thường sử dụng đối tượng của mảng hoặc List hoặc đối tượng triển khai interface IList.
* Code mẫu trong bài học:

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace LearnWindowForm

{

  public partial class learnListBox : Form

  {

      public learnListBox()

      {

          InitializeComponent();

          var brands = new string[] { "Apple", "Samsung", "Xiaomi", "Huawei", "Oppo", "Vivo" };

          var listBrands = new List<string>(brands);

          // listBoxBrands.Items.AddRange(brands);

          // listBoxBrands.SelectedIndex = -1;

          listBoxBrands.DataSource = listBrands;

      }

        private void button1\_Click(object sender, EventArgs e)

      {

          var str = "";

          var selectedItems = listBoxBrands.SelectedItems;

          foreach (var item in selectedItems)

          {

              str += item.ToString() + "\n";

          }

          MessageBox.Show(str);

      }

  }

}

# Bài 2.9. PictureBox

# Video:

<https://youtu.be/CupBM_5XzdQ>

#### **Các điểm cần lưu ý**

* PictureBox là control dùng để hiển thị ảnh. Định dạng ảnh support gồm .bitmap, .jpg, .ico, .bmp, .gif, .wmf.
* Để thiết lập ảnh ta sử dụng thuộc tính Image.
* Để thay đổi kích thước ảnh với các mode có sẵn ta dùng SizeMode.
* Code mẫu thiết lập size mode và ảnh cho PictureBox khi chạy chương trình:

using System;

using System.Drawing;

using System.Windows.Forms;

namespace LearnWindowForm

{

  public partial class LearnPictureBox : Form

  {

      public LearnPictureBox()

      {

          InitializeComponent();

      }

      private void SelectImageChangedHandler(object sender, EventArgs e)

      {

          var imageName = comboBoxSelectedImage.Text;

          var folderPath = @"C:\Windows\Web\Wallpaper\ThemeD";

          var fullPath = folderPath + @"\" + imageName;

          var image = new Bitmap(fullPath);

          pictureBoxWallpaper.Image = image;

      }

        private void SizeModeChangedHandler(object sender, EventArgs e)

      {

          var sizeMode = comboBoxSizeMode.Text;

          if (sizeMode.CompareTo("AutoSize") == 0)

          {

              pictureBoxWallpaper.SizeMode = PictureBoxSizeMode.AutoSize;

          }

          else if (sizeMode.CompareTo("CenterImage") == 0)

          {

              pictureBoxWallpaper.SizeMode = PictureBoxSizeMode.CenterImage;

          }

          else if (sizeMode.CompareTo("Normal") == 0)

          {

              pictureBoxWallpaper.SizeMode = PictureBoxSizeMode.Normal;

          }

          else if (sizeMode.CompareTo("StretchImage") == 0)

          {

              pictureBoxWallpaper.SizeMode = PictureBoxSizeMode.StretchImage;

          }

          else if (sizeMode.CompareTo("Zoom") == 0)

          {

              pictureBoxWallpaper.SizeMode = PictureBoxSizeMode.Zoom;

          }

      }

  }

}

# Bài 2.10. ListView

Video:

<https://youtu.be/BGGNgy7AIro>

#### **Các điểm cần lưu ý**

* ListView cho phép hiển thị các phần tử của tập hợp ở dạng phức tạp gồm cả icon và text như trong FileExplorer.
* Để thiết lập chế độ hiển thị ta thay đổi giá trị của thuộc tính View.
* Để thay đổi chế độ cho phép chọn 1 hoặc nhiều phần tử ta sửa giá trị của MultiSelect.
* Để thêm bớt, sửa đổi phần tử trong ListView ta sử dụng thuộc tính Items.
* Để sửa đổi các cột của ListView ta sử dụng thuộc tính Columns.
* Để sửa đổi các nhóm phần tử ta sử dụng thuộc tính Groups.
* Trong quá trình sửa đổi các phần tử ta có thể thiết lập nhóm, đặt tên, thiết lập độ rộng cột, số cột và các giá trị từng cột thông qua SubItems.
* Click vào ListView và nhìn góc phải dưới màn hình trong cửa sổ Properties có 3 đường dẫn tắt cho phép ta sửa nhanh các thành phần: Items, Columns và Groups.

# Bài 2.11. ColorDialog

Video:

<https://youtu.be/kf5PmhU6394>

#### **Các điểm cần lưu ý**

* ColorDialog là hộp thoại cho phép chọn màu có sẵn hoặc màu cá nhân hóa của bạn.
* Để hiển thị hộp thoại ta gọi ShowDialog().
* Để cho phép hiển thị đầy đủ bảng màu ta thiết lập giá trị của thuộc tính FullOpen là true.
* Để cho phép người dùng chọn màu cá nhân hóa ta tùy chỉnh thuộc tính AllowFullOpen là true.
* Để lấy màu đã chọn ta gọi thuộc tính Color của đối tượng ColorDialog.
* Ví dụ:

using System;

using System.Windows.Forms;

namespace LearnWindowForm

{

  public partial class LearnColorDialogForm : Form

  {

      public LearnColorDialogForm()

      {

          InitializeComponent();

      }

      private void ChangeColorHandler(object sender, EventArgs e)

      {

          colorDialog.ShowDialog();

          var selectedColor = colorDialog.Color;

          this.BackColor = selectedColor;

      }

  }

}

// 1 kết quả chọn màu:

Graphical user interface

Description automatically generated

# Bài 2.12. DateTimePicker

Video:

<https://youtu.be/xsVVj2zfwbk>

#### **Những điểm cần lưu ý**

* DateTimePicker cho phép ta hiển thị, lựa chọn thời gian, ngày tháng.
* Để thiết lập, lấy giá trị thời gian đang hiển thị trong DateTimePicker, sử dụng thuộc tính Value.
* Để thay đổi cách thức hiển thị thông tin thời gian, ngày giờ ta thay đổi thuộc tính Format.
* Để thiết lập định dạng hiển thị ngày giờ theo mong muốn của bạn, sử dụng thuộc tính CustomFormat và Format.
* Để lấy text hiển thị trong DateTimePicker ta sử dụng thuộc tính Text.
* Để thiết lập khoảng ngày tháng min, max hiển thị trong DateTimePicker, ta sử dụng thuộc tính MinDate, MaxDate và thiết lập giá trị cho nó.
* Ví dụ sau trích xuất từ bài học:

using System;

using System.Windows.Forms;

namespace LearnWindowForm

{

  public partial class LearnDateTimePickerForm : Form

  {

      public LearnDateTimePickerForm()

      {

          InitializeComponent();

          dateTimePicker.Value = new DateTime(2025, 1, 1, 15, 34, 26);

      }

        private void DateSelectedHandler(object sender, EventArgs e)

      {

          var selectedDateTime = dateTimePicker.Value; // lấy ngày giờ từ DateTimePicker

          var dayOfWeek = selectedDateTime.DayOfWeek;

          // MessageBox.Show("Day of week: " + dayOfWeek);

          var text = dateTimePicker.Text;

          MessageBox.Show(text); // hiển thị text lấy từ DateTimePicker

      }

  }

}

# Bài 2.13. DataGridView

Video:

<https://youtu.be/OTY4zm_iVE0>

#### **Các điểm cần lưu ý**

* DataGridView thường sử dụng để hiển thị và cho phép thao tác trực tiếp trên dữ liệu ở dạng bảng.
* DataGridView có thể được liên kết với nguồn cấp dữ liệu đọc ra từ file, internet, Database.
* Để thêm, sửa các cột ta chuột phải vào DataGridView và chọn Edit.. hoặc Add.. tương ứng.
* Để căn chỉnh tiêu đề vào giữa cột, sửa thuộc tính ColumnHeadersDefaultCellStyle -> thiết lập Alignment là MiddleCenter.
* Để sửa font, kích thước chữ, style chữ của tiêu đề ta sửa mục Font của ColumnHeadersDefaultCellStyle.
* Để sửa font, kích thước chữ, style, màu chữ… của dữ liệu trong từng ô của bảng, ta thao tác với thuộc tính DefaultCellStyle.
* Để các cột tự động căn chỉnh khớp với độ rộng của giao diện, ta sửa thuộc tính AutoSizeColumnMode về Fill.
* Để thay đổi kiểu dữ liệu trên từng cột ta sửa mục Type khi thêm cột hoặc sửa mục ColumnType khi sửa cột.
* Khi muốn thiết lập tương tác cho từng ô ta phải chỉ định rõ từng vị trí hàng, cột để bắt sự kiện và xử lý tương ứng.
* Để đổ dữ liệu lên DataGridView ta tạo một mảng các object, mỗi phần tử là dữ liệu của một cột tương ứng cho hàng sau đó thêm vào thuộc tính Rows qua phương thức Add() của đối tượng DataGridView().
* Khi muốn update, xóa ta cũng có thể chỉ định rõ chỉ số của hàng cần xóa qua thuộc tính Rows.
* Ví dụ minh họa:

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace LearnWindowForm

{

  public partial class LearnDataGridViewForm : Form

  {

      public List<Student> Students { get; set; } = new List<Student>();

      public LearnDataGridViewForm()

      {

          InitializeComponent();

          CreateFakeData();

      }

      private void CreateFakeData()

      {

          Students.Add(new Student("SV1001", "Trần Trung Đức", "CNTT", new DateTime(2009, 10, 15), 3.15f));

          Students.Add(new Student("SV1002", "Lê Văn Hùng", "CNTT", new DateTime(2009, 3, 17), 3.45f));

          Students.Add(new Student("SV1003", "Ma văn Thắng", "CNTT", new DateTime(2009, 9, 25), 3.55f));

          Students.Add(new Student("SV1004", "Trần Ngô Mỹ Duyên", "CNTT", new DateTime(2009, 12, 11), 3.35f));

          Students.Add(new Student("SV1005", "Lê Hoàng Diệp Thảo", "CNTT", new DateTime(2009, 1, 14), 3.65f));

          Students.Add(new Student("SV1006", "Nguyễn Mai Trần Long", "CNTT", new DateTime(2009, 11, 15), 3.25f));

          Students.Add(new Student("SV1007", "Hoàng Đức Sơn", "CNTT", new DateTime(2009, 4, 27), 3.75f));

          Students.Add(new Student("SV1008", "Nông Văn Tấn", "CNTT", new DateTime(2009, 2, 18), 3.05f));

          Students.Add(new Student("SV1009", "Chu Đình Dũng", "CNTT", new DateTime(2009, 10, 30), 3.15f));

          Students.Add(new Student("SV1010", "Ma Thùy Quyên", "CNTT", new DateTime(2009, 8, 28), 3.45f));

      }

      private void FillData()

      {

          dataGridViewStudent.Rows.Clear();

          foreach (var item in Students)

          {

              dataGridViewStudent.Rows.Add(item.ToPropertiesArray());

          }

      }

      private void FillDataHandler(object sender, EventArgs e)

      {

          FillData();

      }

      private void UpdateTableDataHandler(object sender, DataGridViewCellEventArgs e)

      {

          if (e.ColumnIndex == dataGridViewStudent.Columns["colRemove"].Index &&

              e.RowIndex != -1)

          {

              var row = dataGridViewStudent.Rows[e.RowIndex];

              var id = row.Cells["colId"].Value.ToString();

              var indexToRemove = FindIndex(id);

              if (indexToRemove > -1)

              {

                  var confirmAns = MessageBox.Show("Bạn có chắc chắn muốn xóa bản ghi này không?",

                      "Xóa bản ghi", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

                  if (confirmAns == DialogResult.Yes)

                  {

                      Students.RemoveAt(indexToRemove);

                      // FillData(); // làm mới lại dữ liệu trong bảng

                      // nếu xóa qua phương thức RemoveAt(index) hoặc Remove(object) thì không cần load lại

                      dataGridViewStudent.Rows.RemoveAt(indexToRemove);

                      MessageBox.Show($"Xóa thành công sinh viên mã \"{id}\"!",

                          "Kết quả xóa", MessageBoxButtons.OK, MessageBoxIcon.Information);

                  }

              }

          }

      }

      // phương thức tìm vị trí của đối tượng cần xóa theo mã trong danh sách gốc

      private int FindIndex(string id)

      {

          for (int i = 0; i < Students.Count; i++)

          {

              if (id.CompareTo(Students[i].Id) == 0)

              {

                  return i;

              }

          }

          return -1;

      }

  }

}

Graphical user interface, table

Description automatically generated

# Bài 2.14. SaveFileDialog

Video:

<https://youtu.be/tr-eL1gXgl0>

Những điểm cần lưu ý

* SaveFileDialog cho phép ta chọn nơi lưu và mở file trong Windows.
* Để thiết lập tên file mặc định ta cung cấp dữ liệu cho thuộc tính FileName.
* Để thiết lập đuôi file mặc định ta chỉ định trong thuộc tính DefaultExt.
* Để chỉ định dạng đuôi mở rộng ta sửa dữ liệu của thuộc tính Filter.
* Để thiết lập tiêu đề ta sửa dữ liệu cho thuộc tính Title.
* Để mở hộp thoại ta gọi ShowDialog().
* Để mở file đã chọn ta gọi OpenFile(). Phương thức này trả về một Stream từ đó có thể dùng để đọc ghi file.
* Ví dụ trong bài học:

using System;

using System.IO;

using System.Windows.Forms;

namespace LearnWindowForm

{

  public partial class LearnSaveFileDialogForm : Form

  {

      public LearnSaveFileDialogForm()

      {

          InitializeComponent();

      }

        private void SaveFileHandler(object sender, EventArgs e)

      {

          if(saveFileDialog.ShowDialog() == DialogResult.OK)

          {

              saveFileDialog.Title = "Lưu file mới";

              var text = textBox1.Text;

              var stream = saveFileDialog.OpenFile();

              var sw = new StreamWriter(stream);

              sw.Write(text);

              sw.Close();

              stream.Close();

              MessageBox.Show("Ghi file thành công!", "Thông báo",

                  MessageBoxButtons.OK, MessageBoxIcon.Information);

          }

      }

  }

}

# Bài 2.15. ProgressBar

Video:

<https://youtu.be/qUSGbe-XMMg>

#### **Những điểm cần lưu ý**

* ProgressBar sử dụng để giám sát quá trình thực hiện một hành động cần thời gian dài để hoàn tất. Ví dụ quá trình tải file từ internet về máy, load project, sao chép file lớn…
* Để thiết lập giá trị tối thiểu cần giám sát ta thay đổi giá trị thuộc tính Minimum.
* Để thiết lập giá trị tối đa cần giám sát ta thay đổi giá trị thuộc tính Maximum.
* Để cập nhật giá trị hiện tại ta sử dụng thuộc tính Value.
* Để thiết lập bước thay đổi khi cập nhật giao diện ProgressBar ta thay đổi giá trị thuộc tính Step.
* Để cập nhật ProgressBar ta có thể gọi PerformStep() hoặc gán trực tiếp giá trị vào thuộc tính Value.
* Ví dụ sau minh họa bài học:

namespace LearnWindowForm

{

  partial class LearnProgressBarForm

  {

      /// <summary>

      /// Required designer variable.

      /// </summary>

      private System.ComponentModel.IContainer components = null;

      /// <summary>

      /// Clean up any resources being used.

      /// </summary>

      /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

      protected override void Dispose(bool disposing)

      {

          if (disposing && (components != null))

          {

              components.Dispose();

          }

          base.Dispose(disposing);

      }

      #region Windows Form Designer generated code

      /// <summary>

      /// Required method for Designer support - do not modify

      /// the contents of this method with the code editor.

      /// </summary>

      private void InitializeComponent()

      {

          this.progressBar = new System.Windows.Forms.ProgressBar();

          this.label1 = new System.Windows.Forms.Label();

          this.button1 = new System.Windows.Forms.Button();

          this.SuspendLayout();

          //

          // progressBar

          //

          this.progressBar.Location = new System.Drawing.Point(64, 75);

          this.progressBar.Name = "progressBar";

          this.progressBar.Size = new System.Drawing.Size(495, 43);

          this.progressBar.TabIndex = 0;

          //

          // label1

          //

          this.label1.AutoSize = true;

          this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

          this.label1.Location = new System.Drawing.Point(61, 45);

          this.label1.Name = "label1";

          this.label1.Size = new System.Drawing.Size(135, 20);

          this.label1.TabIndex = 1;

          this.label1.Text = "Tiến độ công việc:";

          //

          // button1

          //

          this.button1.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

          this.button1.Location = new System.Drawing.Point(221, 153);

          this.button1.Name = "button1";

          this.button1.Size = new System.Drawing.Size(163, 43);

          this.button1.TabIndex = 2;

          this.button1.Text = "Bắt đầu";

          this.button1.UseVisualStyleBackColor = true;

          this.button1.Click += new System.EventHandler(this.StartPerformedHandler);

          //

          // LearnProgressBarForm

          //

          this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

          this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

          this.ClientSize = new System.Drawing.Size(635, 237);

          this.Controls.Add(this.button1);

          this.Controls.Add(this.label1);

          this.Controls.Add(this.progressBar);

          this.Name = "LearnProgressBarForm";

          this.Text = "LearnProgressBarForm";

          this.ResumeLayout(false);

          this.PerformLayout();

      }

      #endregion

      private System.Windows.Forms.ProgressBar progressBar;

      private System.Windows.Forms.Label label1;

      private System.Windows.Forms.Button button1;

  }

}

using System;

using System.Threading;

using System.Windows.Forms;

namespace LearnWindowForm

{

  public partial class LearnProgressBarForm : Form

  {

      public LearnProgressBarForm()

      {

          InitializeComponent();

      }

      private void StartPerformedHandler(object sender, EventArgs e)

      {

          progressBar.Value = 0;

          button1.Enabled = false;

          new Thread(() => PerformedAction()) { IsBackground = true }.Start();

      }

      private void PerformedAction()

      {

          while(progressBar.Value < progressBar.Maximum)

          {

              progressBar.PerformStep();

              Thread.Sleep(1000);

          }

          button1.Enabled=true;

      }

  }

}

# Bài 2.16. Cá nhân hóa các controls

Video:

<https://youtu.be/oxrHba3UhC8>

#### **Những điểm cần lưu ý**

* Tạo project loại WindowsFromsControlLibrary(.NET Framework).
* Build project này sau đó tìm file .dll trong mục Debug và thêm vào project cần sử dụng.
* Control sẽ hiển thị ở góc trái trên trong Toolbox. Kéo vào form và sử dụng như bình thường.
* Trong control tự cá nhân hóa có thể định nghĩa tất cả các thuộc tính, phương thức, sự kiện mong muốn và sử dụng sau.
* Tải file thư viện mẫu: <https://drive.google.com/drive/folders/1zTJpoCWigQ568MDAG9Q2IRfy0qrBK6Ds?usp=sharing>

# Bài 2.17. Liên kết dữ liệu trong windows forms

Video:

<https://youtu.be/rXIHNFv-mTE>

#### **Lưu ý**

* Có nhiều cách liên kết dữ liệu vào bảng. Cách thứ nhất là liên kết trực tiếp với bảng trong Database.
* Cách thứ hai là đọc dữ liệu từ file, object, database sau đó gắn vào DataSource.
* Dữ liệu sử dụng bạn lấy từ các bài tập và bài học trong chương 9.
* Code mẫu cách 2:

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace LearnWindowForm

{

  public partial class BindingDataGridviewForm : Form

  {

      private DataTable dt = new DataTable();

      public BindingDataGridviewForm()

      {

          InitializeComponent();

      }

      private void LoadDataHandler(object sender, EventArgs e)

      {

          var connString = "Data Source=BraniumAcademy;Initial Catalog=LearnCSharp;User ID=sa;Password=123";

          var sql = $"SELECT \* FROM Student WHERE FullName LIKE '%t%'";

          var conn = new SqlConnection(connString);

          conn.Open();

          var command = new SqlCommand(sql, conn);

          using (var reader = command.ExecuteReader())

          {

              if (reader.HasRows)

              {

                  dt.Load(reader);

              }

              dataGridView1.DataSource = dt;

          }

          conn.Close();

      }

  }

}

# Bài 2.18. Xuất bản ứng dụng và cài đặt nó vào máy tính khác

Video:

<https://youtu.be/4sVWQ3HnsX4>

# Bài 2.19. Sử dụng menu

Video:

<https://youtu.be/9cOkaBv9FnI>

# Bài 13.20. Ví dụ tổng hợp

#### **Giới thiệu**

<https://youtu.be/2e1AqAmX21E>

#### **Thiết kế giao diện cha**

<https://youtu.be/-xDLBO2ui48>

#### **Thiết kế giao diện con**

<https://youtu.be/12ZhKPyDvVM>

#### **Thêm dữ liệu và xử lý các ràng buộc**

<https://youtu.be/DM0rS16u_sw>

#### **Cập nhật thông tin sinh viên**

<https://youtu.be/wNhkhmGTouM>

#### **Đọc ghi file thông tin sinh viên**

<https://youtu.be/BJ0gulZQuTE>

#### **Thực hiện các chức năng sắp xếp**

<https://youtu.be/mLXC9l16-3k>

#### **Thực hiện các chức năng tìm kiếm**

<https://youtu.be/mT5grr2SfQY>

#### **Xóa thông tin sinh viên khỏi bảng**

<https://youtu.be/x4Q19oRhpJM>

**CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG QUẢN LÝ BÁN HÀNG SUPER MALL**

# Bài 3.0. Yêu cầu dự án – Project requirement

# <https://drive.google.com/file/d/1D_5ahSxlUTZWlPc5yZ7M-r84oVeLtHqv/view?usp=sharing>

# Bài 3.1. Giới thiệu các chức năng của ứng dụng

<https://youtu.be/CS76Osy_p74>

# Bài 3.2. Thiết kế tổng quan toàn hệ thống

<https://youtu.be/sw-WD2O_5PQ>

# Bài 3.3. Phân tích thiết kế các lớp thực thể p1

<https://youtu.be/JdGWFz2OG7g>

# Bài 3.4. Phân tích thiết kế các lớp thực thể p2

<https://youtu.be/IvcdOGip8KA>

# Bài 3.5. Thiết kế giao diện chức năng quản lý mặt hàng

<https://youtu.be/_NQtoEgHJ4Y>

# Bài 3.6. Thiết kế giao diện quản lý khách hàng

<https://youtu.be/hUaKMUOLGWQ>

# Bài 3.7. Thiết kế giao diện quản lý khuyến mãi

<https://youtu.be/LMn3R30eFAs>

# Bài 3.8. Thiết kế giao diện quản lý đơn hàng và thống kê

<https://youtu.be/r86cj2_lFIc>

# Bài 3.9. Thiết kế giao diện thêm/sửa mặt hàng

<https://youtu.be/lXtS0Yv49gk>

# Bài 3.10. Thêm mới mặt hàng

<https://youtu.be/V6MH30Va8AM>

# Bài 3.11. Sửa thông tin mặt hàng

<https://youtu.be/x6NbPAKpMiI>

# Bài 3.12. Xóa thông tin mặt hàng

<https://youtu.be/6SE898_IrMk>

# Bài 3.13. Sắp xếp thông tin mặt hàng

<https://youtu.be/zp85-2TnTTI>

# Bài 3.14. Tìm kiếm mặt hàng

<https://youtu.be/wdj2CntA_q0>

# Bài 3.15. Sửa/xóa mặt hàng sau khi tìm kiếm

<https://youtu.be/jrP2FeFdgGY>

# Bài 3.16. Thiết kế giao diện thêm mới khách hàng

<https://youtu.be/8FqIbrUm_Bk>

# Bài 3.17. Thêm mới khách hàng

<https://youtu.be/JP7eiCokUUw>

# Bài 3.18. Sửa + xóa thông tin khách hàng

<https://youtu.be/5E6PaQtZ6ro>

# Bài 3.19. Sắp xếp thông tin khách hàng

<https://youtu.be/3JgtINJOOo8>

# Bài 3.20. Tìm kiếm thông tin khách hàng

<https://youtu.be/bBmFZEbmRnY>

# Bài 3.21. Thiết kế giao diện thêm mới khuyến mãi

<https://youtu.be/h1GTV_7Tcz4>

# Bài 3.22. Thêm mới chương trình khuyến mãi

<https://youtu.be/hoVXQzuW26U>

# Bài 3.23. Sửa + xóa thông tin khuyến mãi

<https://youtu.be/TE_PuDbYd3Y>

# Bài 3.24. Tìm kiếm thông tin khuyến mãi

<https://youtu.be/WUxXXT7v3-I>

# Bài 3.25. Gắn khuyến mãi cho mặt hàng

<https://youtu.be/7FVNVAO8iO0>

# Bài 3.26. Thiết kế giao diện tạo mới hóa đơn

<https://youtu.be/DYSsDW5snHY>

# Bài 3.27. Thiết kế giao diện thanh toán

<https://youtu.be/9X7kzTDVulg>

# Bài 3.28. Thêm mới, sửa, xóa hóa đơn và fix lỗi

<https://youtu.be/JlP7SXnKrxE>

<https://youtu.be/eQ82WzY31aM>

<https://youtu.be/G-XL9gBuaCA>

<https://youtu.be/wFgI_GrXctk>

<https://youtu.be/pYbXIO4pBNs>

<https://youtu.be/j6TJaJv7ZQE>

<https://youtu.be/zn-NMayo4Kk>

<https://youtu.be/jKQGvgq5RBo>

# Bài 3.29. Thực hiện chức năng lưu file JSON

<https://youtu.be/46Um0VuJ6KQ>

# Bài 3.30. Cập nhật mã tự động tăng sau khi đọc dữ liệu từ File/CSDL

<https://youtu.be/Vui3tff3sJc>

# Bài 3.31. Thực hiện chức năng sắp xếp danh sách hóa đơn

<https://youtu.be/8xxvCJuF9ss>

# Bài 3.32. Thực hiện chức năng tìm kiếm hóa đơn

<https://youtu.be/_SMZScYCOLk>

# Bài 3.33. Thiết kế giao diện chức năng thống kê

<https://youtu.be/EY6vISdpghM>

# Bài 3.34. Thống kê các mặt hàng có doanh thu cao nhất

<https://youtu.be/69wWWK2fvKE>

*Ý tưởng:*

* Chỉ xét các hóa đơn đã thanh toán.
* Xét từng mặt hàng được chọn trong danh sách các mặt hàng có trong giỏ hàng của từng đơn hàng.
* Nếu sản phẩm đã tồn tại trong danh sách kết quả: cộng dồn số lượng và tổng tiền của sản phẩm loại đó vào mặt hàng trong danh sách kết quả thống kế.
* Nếu sản phẩm chưa tồn tại trong danh sách kết quả thống kê: thêm sản phẩm, tổng số sản phầm và tổng tiền của sản phẩm đó trong đơn hàng đang xét vào danh sách kết quả thống kê.
* Sau cùng sắp xếp các sản phẩm thống kê giảm dần theo tổng doanh thu.
* Hiển thị kết quả lên bảng.

# Bài 3.35. Thống kê các khách hàng mua nhiều nhất

<https://youtu.be/5We7f8QRoZg>

# Bài 3.36. Thống kê top 10 ngày trong tháng có doanh thu cao nhất

<https://youtu.be/VVZ08T-t2CA>

# Bài 3.37. Thống kê doanh thu theo tháng

<https://youtu.be/ekAsS9eiMJ8>

# Bài 3.38. Thống kê doanh thu theo ngày

<https://youtu.be/jjSUXVK5ULQ>

# Bài 3.39. Thêm tính năng cho menu

<https://youtu.be/C0yvtctPiBA>

=====================================Hết==================================