Tutorial 18

Programming exercise 1 – Course edit form

1. Add new file to WIUT project and declare enumeration FormMode

public enum FormMode

{

CreateNew,

Update

}

1. Add a new form, call it CourseEditForm
2. Change title of the form to Course
3. Add corresponding controls to the form – label and textbox for name, buttons Save and Cancel
4. Change Tab Order for the form to ensure proper navigation
5. Add the following code to the form

public Course Course { get; set; }

public FormMode Mode { get; set; }

public void CreateNewCourse()

{

Mode = FormMode.CreateNew;

Course = new Course();

InitializeControls();

MdiParent = MyForms.GetForm<ParentForm>();

Show();

}

public void UpdateCourse(Course course)

{

Mode = FormMode.Update;

Course = course;

InitializeControls();

ShowCourseInControls();

MdiParent = MyForms.GetForm<ParentForm>();

Show();

}

private void InitializeControls()

{

}

private void ShowCourseInControls()

{

}

private void GrabUserInput()

{

}

1. Add message box for Save button click event

MessageBox.Show("Save clicked");

1. Add message box for Cancel button click event

Close();

1. Set AcceptButton property of the form to Save button, CancelButton property to Cancel button
2. Open ParentForm and create a handler for Add new course menu item. Add the following code:

new CourseEditForm().CreateNewCourse();

1. Run the project and click on Add new course menu item. Click on Enter and Esc buttons and see the result
2. Populate ShowCourseInControls() and GrabUserInput() subs:

private void ShowCourseInControls()

{

tbxName.Text = Course.Name;

}

private void GrabUserInput()

{

Course.Name = tbxName.Text;

}

1. Go to CourseListForm and make function LoadData() public
2. Populate Save button click handlers

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

{

try

{

GrabUserInput();

var manager = new CourseManager();

if (Mode == FormMode.CreateNew)

manager.Create(Course);

else

manager.Update(Course);

MyForms.GetForm<CourseListForm>().LoadData();

Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

1. Run the project and see the result
2. Open CourseListForm. Set MultiSelect property of the DataGridView to False.
3. Create handlers for Add, Update and Delete buttons

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

{

new CourseEditForm().CreateNewCourse();

}

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

{

if (dgv.SelectedRows.Count == 0)

MessageBox.Show("Please select a course");

else

{

var c = (Course)dgv.SelectedRows[0].DataBoundItem;

new CourseEditForm().UpdateCourse(c);

}

}

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

{

if (dgv.SelectedRows.Count == 0)

MessageBox.Show("Please select a course");

else

{

var c = (Course)dgv.SelectedRows[0].DataBoundItem;

new CourseManager().Delete(c.Id);

LoadData();

}

}

1. Run the project and see the result

Home work

Create applicant edit form

You can use the following code as foothold:

public Applicant Applicant { get; set; }

public FormMode Mode { get; set; }

public void CreateNewApplicant()

{

Mode = FormMode.CreateNew;

Applicant = new Applicant();

InitializeControls();

MdiParent = MyForms.GetForm<ParentForm>();

Show();

}

public void UpdateApplicant(Applicant applicant)

{

Mode = FormMode.Update;

Applicant = applicant;

InitializeControls();

ShowApplicantInControls();

MdiParent = MyForms.GetForm<ParentForm>();

Show();

}

private void InitializeControls()

{

cbxCourse.DataSource = new CourseManager().GetAll();

}

private void ShowApplicantInControls()

{

}

private void GrabUserInput()

{

}