4. In the code editor, click in the blank line below the comment, and then type the following code:

```
public event EventHandler LogonSuccess;
```

- 5. In the Task List window, double-click the TODO: Exercise 1: Task 2b: Implement the Logon\_Click event handler for the Logon button task.
- 6. In the code editor, click in the blank line below the comments, and then type the following code:

#### delegate

## How would you explain this?

4. In the code editor, click in the blank line below the comment, and then type the following code:

public event EventHandler LogonSuccess;

#### Declare an event of type EventHandler

- In the Task List window, double-click the TODO: Exercise 1: Task 2b: Implement the Logon\_Click event handler for the Logon button task.
- 6. In the code editor, click in the blank line below the comments, and then type the following code:

```
private void Logon_Click(object sender, RoutedEventArgs e)
{
    // Save the username and role (type of user) specified on the form in the global context
    SessionContext.UserName = username.Text;
    SessionContext.UserRole = (bool)userrole.IsChecked ? Role.Teacher : Role.Student;

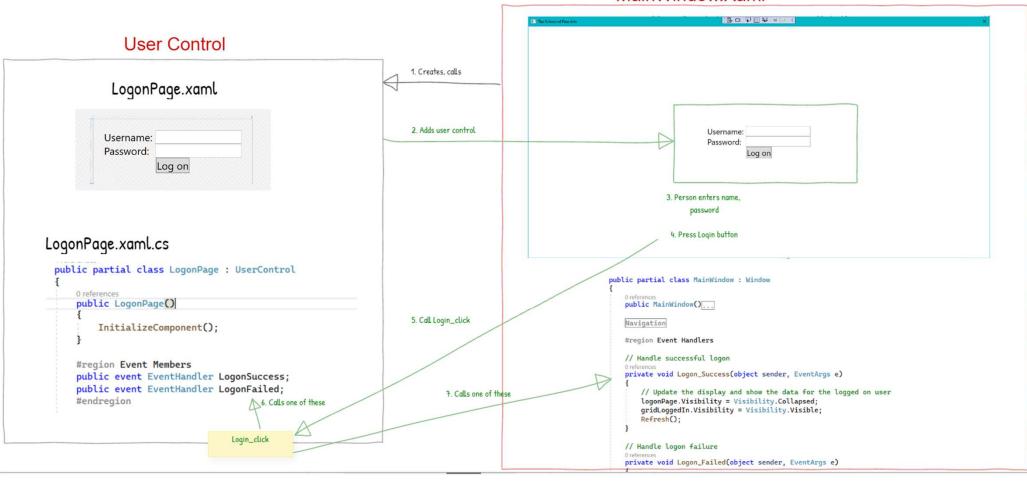
    // If the role is Student, set the CurrentStudent property in the global context to a dummy student; Eric if (SessionContext.UserRole == Role.Student)
    {
        SessionContext.CurrentStudent = "Eric Gruber";
    }
}
```

```
// Raise the LogonSuccess event
if (LogonSuccess != null)
{
    LogonSuccess(this, null);
}
```

As long as something is subscribing to this event

Call the Event

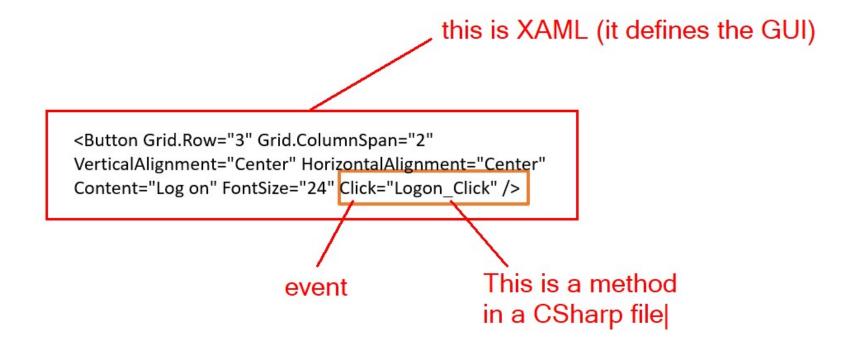
#### MainWindow.Xaml

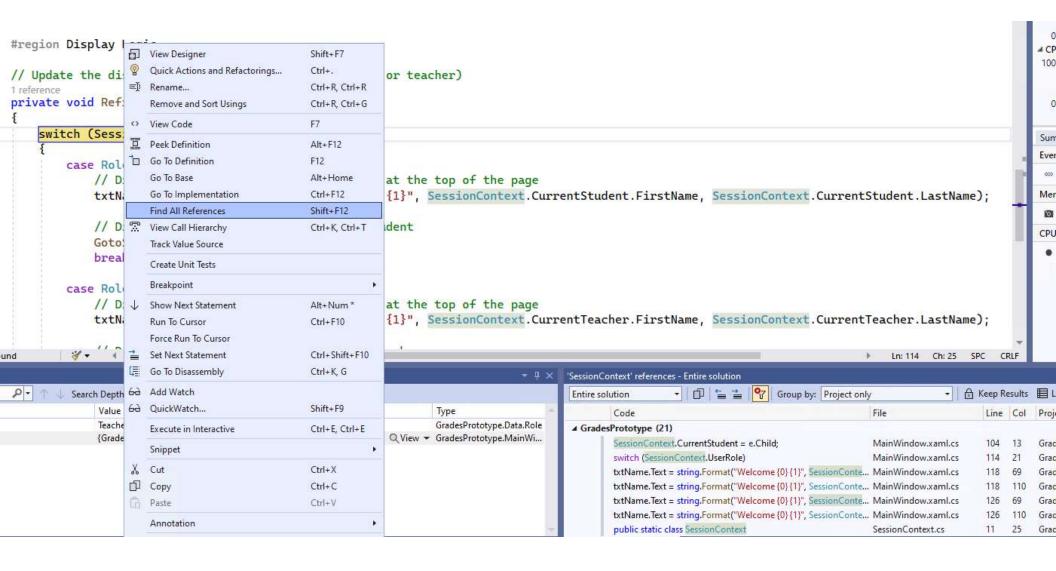


```
<Button Grid.Row="3" Grid.ColumnSpan="2"

VerticalAlignment="Center" HorizontalAlignment="Center"

Content="Log on" FontSize="24" Click="Logon_Click" />
```





```
// Handle successful logon
private void Logon_Success(object sender, EventArgs e)
{
    // Update the display and show the data for the logged on user
    logonPage.Visibility = Visibility.Collapsed;
    gridLoggedIn.Visibility = Visibility.Visible;
    Refresh();
}
```

```
<y:LogonPage x:Name="logonPage" LogonSuccess="Logon_Success" Visibility="Collapsed" />
```

only accessible in this class (or struct)

#### What's this? Where did it come from?

```
switch
{
    case Role.Student:
        // Display the student name in the banner at the top of the page
        txtName.Text = string.Format("Welcome {0}", SessionContext.UserName);

        // Display the details for the current student
        GotoStudentProfile();
        break;

case Role.Teacher:
        // Display the teacher name in the banner at the top of the page
        txtName.Text = string.Format("Welcome {0}", SessionContext.UserName);

        // Display the list of students for the teacher
        GotoStudentsPage();
        break;
}
```

```
// Parse the student name into the first name and last name by using a regular expression
// The firstname is the initial string up to the first space character.
// The lastname is the string after the space character
Match matchNames = Regex.Match(SessionContext.CurrentStudent, @"([^ ]+) ([^ ]+)");
if (matchNames.Success)
    string firstName = matchNames.Groups[1].Value; // Indexing in the Groups collection starts at 1, not 0
    string lastName = matchNames.Groups[2].Value;
    // Display the first name and last name in the TextBlock controls in the studentName StackPanel
    ((TextBlock)studentName.Children[0]).Text = firstName;
    ((TextBlock)studentName.Children[1]).Text = lastName;
// If the current user is a student, hide the Back button
// (only applicable to teachers who can use the Back button to return to the list of students)
if (SessionContext.UserRole == Role.Student)
    btnBack.Visibility = Visibility.Hidden;
else
    btnBack.Visibility = Visibility.Visible;
```

# regex is a class that is useful for finding string and/or replacing strings

```
// Parse the student name into the first name and last name by using a regular expression
// The firstname is the initial string up to the first space character.
// The lastname is the string after the space character
Match matchNames = Regex.Match(SessionContext.CurrentStudent, @"([^ ]+) ([^ ]+)");
if (matchNames.Success)
    string firstName = matchNames.Groups[1].Value; // Indexing in the Group collection starts at 1, not 0
    string lastName = matchNames.Groups[2].Value;
   // Display the first name and last name in the TextBlock controls in the studentName StackPanel
   ((TextBlock)studentName.Children[0]).Text = firstName;
                                                           changing the user interface
   ((TextBlock)studentName.Children[1]).Text = lastName;
// If the current user is a student, hide the Back button
// (only applicable to teachers who can use the Back button to return to the list of students)
if (SessionContext.UserRole == Role.Student)
    btnBack.Visibility = Visibility.Hidden;
else
    btnBack.Visibility = Visibility.Visible;
```

```
Button itemClicked = sender as Button;
if (itemClicked != null)
{
    // Find out which student was clicked - the Tag property of the button contains the name
    string studentName = (string)itemClicked.Tag;
    if (StudentSelected != null)
    {
        // Raise the StudentSelected event (handled by MainWindow) to display the details for this student
        StudentSelected(sender, new StudentEventArgs(studentName));
}
```

```
this is a button

Button itemClicked = sender as Button;

if (itemClicked != sull)

{

// Find out which student was clicked - the Tag property of the button contains the name string studentName = (string)itemClicked.Tag; you can use tag for anything. in this if (StudentSelected != null case it hold a student's name.)

{

// Raise the StudentSelected event (handled by MainWindow) to display the details for this student StudentSelected(sender, new StudentEventArgs(studentName));

}
```

4. In the code editor, click in the blank line in the studentsPage\_StudentSelected method, and then type the following code:

SessionContext.CurrentStudent = e.Child;
GotoStudentProfile();

```
studentsPage.Visibility = Visibility.Collapsed;
studentProfile.Visibility = Visibility.Collapsed;
logonPage.Visibility = Visibility.Visible;
}

this method is

// Handle the Back button on the Student view called via an event

oreferences
private void studentPage_Back(object sender, EventArgs e)

GotoStudentsPage();
}

// Handle the StudentSelected event when the user clicks a student on the Students

oreferences
private void studentsPage_StudentSelected(object sender) StudentEventArgs e)

SessionContext.CurrentStudent = e.Child; simselapsed
GotoStudentProfile();

#region Display Logic

// Undate the display for the logged on user (student or teacher)
```

QuickWatch		A114
Expression:		
e.Child		
Value:		
Name	Value	
▲ 🏂 e.Child	{GradesPrototype.Data.Student}	
FirstName	"Martin"	Q Vie
LastName	"Weber"	Q Vie
Password	"password"	Q Vie
StudentID	2	- 2
TeacherID	1	
<b>№</b> UserName	"weberm"	Q Vie

```
public struct Grade
{
    public int StudentID { get; set; }
    public string AssessmentDate { get; set; }
    public string SubjectName { get; set; }
    public string Assessment { get; set; }
    public string Comments { get; set; }
}
```

6. In the Task List window, double-click the TODO: Exercise 2: Task 1b: Create the Student struct task.

7. In the code editor, click in the blank line below the comment, and then type the following code:

```
public struct StudentID { get; set; }
  public int StudentID { get; set; }
  public string UserName { get; set; }
  public string Password { get; set; }
  public int TeacherID { get; set; }
  public string FirstName { get; set; }
  public string LastName { get; set; }
}
```

8. In the Task List window, double-click the TODO: Exercise 2: Task 1c: Create the Teacher struct task.

9. In the code editor, click at the end of the comment line, press Enter, and then type the following code:

```
public struct Teacher
{
   public int TeacherID { get; set; }
   public string UserName { get; set; }
   public string Password { get; set; }
   public string FirstName { get; set; }
   public string LastName { get; set; }
   public string Class { get; set; }
}
```

```
class Program
   static void Main(string[] args)
       Person p1 = new Person();
       p1.Name = "Alek";
       p1.Age = -9;
                                                shorthand way of
struct Person
                                                writing a property
   public string Name { get; set; }
                                             field
   private int _age;
   public int Age
       get { return _age; }
       set {
           if (value < 0)
              throw new InvalidProgramException();
           _age = value;
```

C# Linq expression.
A bit like SQL

```
// Find all the grades for the student
ArrayList grades = new ArrayList();

foreach (Grade grade in DataSource.Grades)
{
    if (grade.StudentID == SessionContext.CurrentStudent.StudentID)
    {
        grades.Add(grade);
    }
}

// Display the grades in the studentGrades ItemsControl by using databinding studentGrades.ItemsSource = grades;
```

# A collection that can hold a set of data. It can grow and shrink in size. How would you explain this?

```
// Find all the grades for the student
ArrayList grades = new ArrayList();

foreach (Grade grade in DataSource.Grades)
{
    if (grade.StudentID == SessionContext.CurrentStudent.StudentID)
    {
        grades.Add(grade);
    }
}

// Display the grades in the studentGrades ItemsControl by using databinding studentGrades.ItemsSource = grades;
```

Show it in the user interface