

# 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;
```

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:

```
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);
    }
}
```

# How would you explain this?

delegate

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

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:

```
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

## User Control

### LogonPage.xaml

Username:   
Password:

### LogonPage.xaml.cs

```
public partial class LogonPage : UserControl
{
    0 references
    public LogonPage()
    {
        InitializeComponent();
    }

    #region Event Members
    public event EventHandler LogonSuccess;
    public event EventHandler LogonFailed;
    #endregion

```

Login\_click

## MainWindow.Xaml

Username:   
Password:

3. Person enters name,  
password

4. Press Login button

```
public partial class MainWindow : Window
{
    0 references
    public MainWindow()
    {
        Navigation
    }

    #region Event Handlers

    // Handle successful logon
    0 references
    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();
    }

    // Handle logon failure
    0 references
    private void Logon_Failed(object sender, EventArgs e)
    {
    }
    }

```

1. Creates, calls

2. Adds user control

5. Call Login\_click

7. Calls one of these

6. Calls one of these

# How would you explain this?

```
<Button Grid.Row="3" Grid.ColumnSpan="2"  
VerticalAlignment="Center" HorizontalAlignment="Center"  
Content="Log on" FontSize="24" Click="Logon_Click" />
```

# How would you explain this?

this is XAML (it defines the GUI)

```
<Button Grid.Row="3" Grid.ColumnSpan="2"  
VerticalAlignment="Center" HorizontalAlignment="Center"  
Content="Log on" FontSize="24" Click="Logon_Click" />
```

event

This is a method  
in a CSharp file

#region Display

// Update the di

1 reference  
private void Ref

{

switch (Sess

{

case Rol

// D

txtN

// D

Goto

break

case Rol

// D

txtN

und

Search Depth

Value

Teache

{Grade

	View Designer	Shift+F7
	Quick Actions and Refactorings...	Ctrl+.
	Rename...	Ctrl+R, Ctrl+R
	Remove and Sort Usings	Ctrl+R, Ctrl+G
	View Code	F7
	Peek Definition	Alt+F12
	Go To Definition	F12
	Go To Base	Alt+Home
	Go To Implementation	Ctrl+F12
	Find All References	Shift+F12
	View Call Hierarchy	Ctrl+K, Ctrl+T
	Track Value Source	
	Create Unit Tests	
	Breakpoint	
	Show Next Statement	Alt+Num *
	Run To Cursor	Ctrl+F10
	Force Run To Cursor	
	Set Next Statement	Ctrl+Shift+F10
	Go To Disassembly	Ctrl+K, G
	Add Watch	
	QuickWatch...	Shift+F9
	Execute in Interactive	Ctrl+E, Ctrl+E
	Snippet	
	Cut	Ctrl+X
	Copy	Ctrl+C
	Paste	Ctrl+V
	Annotation	

or teacher)

at the top of the page

{1}", SessionContext.CurrentStudent.FirstName, SessionContext.CurrentStudent.LastName);

ident

at the top of the page

{1}", SessionContext.CurrentTeacher.FirstName, SessionContext.CurrentTeacher.LastName);

Ln: 114 Ch: 25 SPC CRLF

'SessionContext' references - Entire solution

Entire solution	Code	File	Line	Col	Proj
GradesPrototype (21)					
	SessionContext.CurrentStudent = e.Child;	MainWindow.xaml.cs	104	13	Grac
	switch (SessionContext.UserRole)	MainWindow.xaml.cs	114	21	Grac
	txtName.Text = string.Format("Welcome {0} {1}", SessionConte...	MainWindow.xaml.cs	118	69	Grac
	txtName.Text = string.Format("Welcome {0} {1}", SessionConte...	MainWindow.xaml.cs	118	110	Grac
	txtName.Text = string.Format("Welcome {0} {1}", SessionConte...	MainWindow.xaml.cs	126	69	Grac
	txtName.Text = string.Format("Welcome {0} {1}", SessionConte...	MainWindow.xaml.cs	126	110	Grac
	public static class SessionContext	SessionContext.cs	11	25	Grac

# How would you explain this?

```
// 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" />
```



# How would you explain this?

only accessible in this class (or struct)

```
// 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();
}
```

Event

Method

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

+



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

```
switch (SessionContext.UserRole)
{
    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;
}
```

# How would you explain this?

```
// 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

## How would you explain this?

```
// 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, @"([^\s]+) ([^\s]+)");

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 TextBox controls in the studentName StackPanel
    ((TextBox)studentName.Children[0]).Text = firstName;
    ((TextBox)studentName.Children[1]).Text = lastName;
}
```

changing the user interface

```
// 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;
}
```

# How would you explain this?

```
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));
    }
}
```

# How would you explain this?

this is a button

converting the variable 'sender' to a 'button'

```
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));
    }
}
```

you can use tag for anything. in this case it hold a student's name.

# How would you explain this?

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;
}

// Handle the Back button on the Student view
0 references
private void studentPage_Back(object sender, EventArgs e)
{
    GotoStudentsPage();
}

// Handle the StudentSelected event when the user clicks a student on the Students
0 references
private void studentsPage_StudentSelected(object sender, StudentEventArgs e)
{
    SessionContext.CurrentStudent = e.Child; ≤ 1ms elapsed
    GotoStudentProfile();
}
#endregion

#region Display Logic

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

```

this method is  
called via an event

QuickWatch		
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	
✚ TeacherID	1	
✚ UserName	"weberm"	Q Vie



# How would you explain this?

```
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 Student
{
    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;
    }
}
```

```
struct Person
{
    public string Name { get; set; }

    private int _age;

    public int Age
    {
        get { return _age; }
        set {
            if (value < 0)
                throw new InvalidOperationException();
            _age = value;
        }
    }
}
```

shorthand way of  
writing a property

field

Property

# How would you explain this?

```
// Find the user in the list of possible users - first check whether the user is a teacher
var teacher = (from Teacher t in DataSource.Teachers
               where String.Compare(t.UserName, username.Text) == 0 &&
                     String.Compare(t.Password, password.Password) == 0
               select t).FirstOrDefault();

// If the UserName of the user retrieved by using LINQ is non-empty then the user is a teacher
if (!String.IsNullOrEmpty(teacher.UserName))
{
    // Save the UserID and Role (teacher or student) and UserName in the global context
    SessionContext.UserID = teacher.TeacherID;
    SessionContext.UserRole = Role.Teacher;
    SessionContext.UserName = teacher.UserName;
    SessionContext.CurrentTeacher = teacher;

    // Raise the LogonSuccess event and finish
    LogonSuccess(this, null);
    return;
}
```

# How would you explain this?

C# Linq expression.  
A bit like SQL

```
// Find the user in the list of possible users - first check whether the user is a teacher
var teacher = (from Teacher t in DataSource.Teachers
               where String.Compare(t.UserName, username.Text) == 0 &&
                     String.Compare(t.Password, password.Password) == 0
               select t).FirstOrDefault();

// If the UserName of the user retrieved by using LINQ is non-empty then the user is a teacher
if (!String.IsNullOrEmpty(teacher.UserName))
{
    // Save the UserID and Role (teacher or student) and UserName in the global context
    SessionContext.UserID = teacher.TeacherID;
    SessionContext.UserRole = Role.Teacher;
    SessionContext.UserName = teacher.UserName;
    SessionContext.CurrentTeacher = teacher;

    // Raise the LogonSuccess event and finish
    LogonSuccess(this, null);
    return;
}
```

# 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;
```

---

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