## Lab: Writing the Code for the Grades Prototype Application

### Lab Setup

Estimated Time: **90 minutes**

### Exercise 1: Adding Navigation Logic to the Grades Prototype Application

#### Task 1: Examine the window and views in the application

1. Click **Visual Studio 2017**.
2. In **Visual Studio**, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, browse to **E:/Allfiles/Mod03/Labfiles/Starter/Exercise 1**, click **GradesPrototype.sln**, and then click **Open**.

>**Note :** If any Security warning dialog box appears, clear **Ask me for every project in this solution** check box and then click **OK**.

1. On the **Build** menu, click **Build Solution**.
2. In **Solution Explorer**, expand **GradesPrototype**, and then double-click **MainWindow.xaml**.
3. Note that this is the main window for the application that will host the following views:
   * **LogonPage.xaml**
   * **StudentProfile.xaml**
   * **StudentsPage.xaml**
4. In **Solution Explorer**, expand **Views**, and then double-click **LogonPage.xaml**.
5. Notice that this view contains text boxes for the username and password, a check box to identify the user as a teacher, and a button to log on to the application.
6. In **Solution Explorer**, double-click **StudentProfile.xaml**.
7. Notice that this view contains a report card that currently displays a list of dummy grades. The view also contains a **Back** button and a blank space that will display the student’s name. This view is displayed when a student logs on or when a teacher views a student’s profile.
8. In **Solution Explorer**, double-click **StudentsPage.xaml**.
9. Notice that this view contains the list of students in a particular class. This view is displayed when a teacher logs on. A teacher can click a student’s name and the **Students Profile** view will be displayed, containing the selected student’s data.

#### Task 2: Define the LogonSuccess event and add dummy code for the Logon\_Click event

1. On the **View** menu, click **Task List**.
2. In the **Task List** window, choose **Entire Solution** option from the list on the left side.
3. Double-click the **TODO: Exercise 1: Task 2a: Define the LogonSuccess event handler** task.
4. In the code editor, click in the blank line below the comment, and then type the following code:

public event EventHandler LogonSuccess;

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 2b: Implement the Logon\_Click event handler for the Logon button** task.
2. 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 Gruber  
   if (SessionContext.UserRole == Role.Student){  
   SessionContext.CurrentStudent = "Eric Gruber";  
   }  
    
   // Raise the LogonSuccess event  
   if (LogonSuccess != null)  
   {  
   LogonSuccess(this, null);  
   }
* }

1. In **Solution Explorer**, double-click **LogonPage.xaml**.
2. In the XAML editor, locate the task **TODO: Exercise 1: Task 2c: Specify that the Logon button should raise the Logon\_Click event handler in this view** task.
3. In the line below the comment, modify the XAML markup

<Button Grid.Row="3" Grid.ColumnSpan="2" VerticalAlignment="Center"

HorizontalAlignment="Center" Content="Log on" FontSize="24" \>

to look like the following markup:

<Button Grid.Row="3" Grid.ColumnSpan="2" VerticalAlignment="Center"

HorizontalAlignment="Center" Content="Log on" FontSize="24"

Click="Logon\_Click" />

#### Task 3: Add code to display the Log On view

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 3a: Display the logon view and hide the list of students and single student view** task.
2. In the code editor, click in the blank line in the **GotoLogon** method, and then type the following code:

// Display the logon view and hide the list of students and single student view logonPage.Visibility = Visibility.Visible;

studentsPage.Visibility = Visibility.Collapsed;

studentProfile.Visibility = Visibility.Collapsed;

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 3b: Handle successful logon** task.
2. In the code editor, click in the blank line below the comments, and then type the following code:

// 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();

}

1. In **Solution Explorer**, double-click **MainWindow.xaml**.
2. In the XAML editor, locate the task **TODO: Exercise 1: Task 3c: Catch the LogonSuccess event and call the Logon\_Success event handler (to be created)** task.
3. In the line below the comment, modify the XAML markup

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

to look like the following markup:

<y:LogonPage x:Name="logonPage" LogonSuccess="Logon\_Success"

Visibility="Collapsed" />

#### Task 4: Add code to determine the type of user

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 4a: Update the display for the logged on user (student or teacher)** task.
2. In the code editor, click in the blank line in the **Refresh** method, and then type the following code:

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;
* }

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 4b: Display the details for a single student** task.
2. In the code editor, click in the blank line in the **GotoStudentProfile** method, and then type the following code:

// Hide the list of students

studentsPage.Visibility = Visibility.Collapsed;

* // Display the view for a single student student
* studentProfile.Visibility = Visibility.Visible;
* studentProfile.Refresh();

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 4c: Display the list of students** task.
2. In the code editor, click in the blank line in the **GotoStudentsPage** method, and then type the following code:

// Hide the view for a single student (if it is visible)

studentProfile.Visibility = Visibility.Collapsed;

* // Display the list of students students
* studentsPage.Visibility = Visibility.Visible;
* studentsPage.Refresh();

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 4d: Display the details for the current student including the grades for the student** task.
2. In the code editor, click in the blank line in the **Refresh** method, and then type the following code:

// 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;
* }

#### Task 5: Handle the Student\_Click event

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 5a: Handle the click event for a student** task.
2. In the code editor, click in the blank line in the **Student\_Click** method, and then type the following code:

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));

}

}

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 5b: Handle the StudentSelected event when the user clicks a student on the Students page** task.
2. 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();

1. In **Solution Explorer**, double-click **MainWindow.xaml**.
2. In the XAML editor, locate the task **TODO: Exercise 1: Task 5c: Catch the StudentSelected event and call the studentsPage\_StudentSelected event handler** task.
3. In the line below the comment, modify the XAML markup

<y:StudentsPage x:Name="studentsPage" Visibility="Collapsed" />

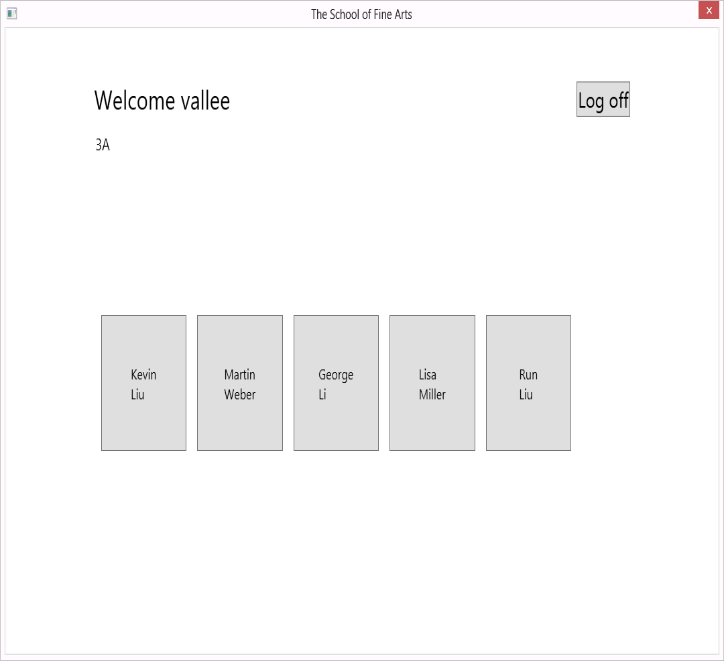
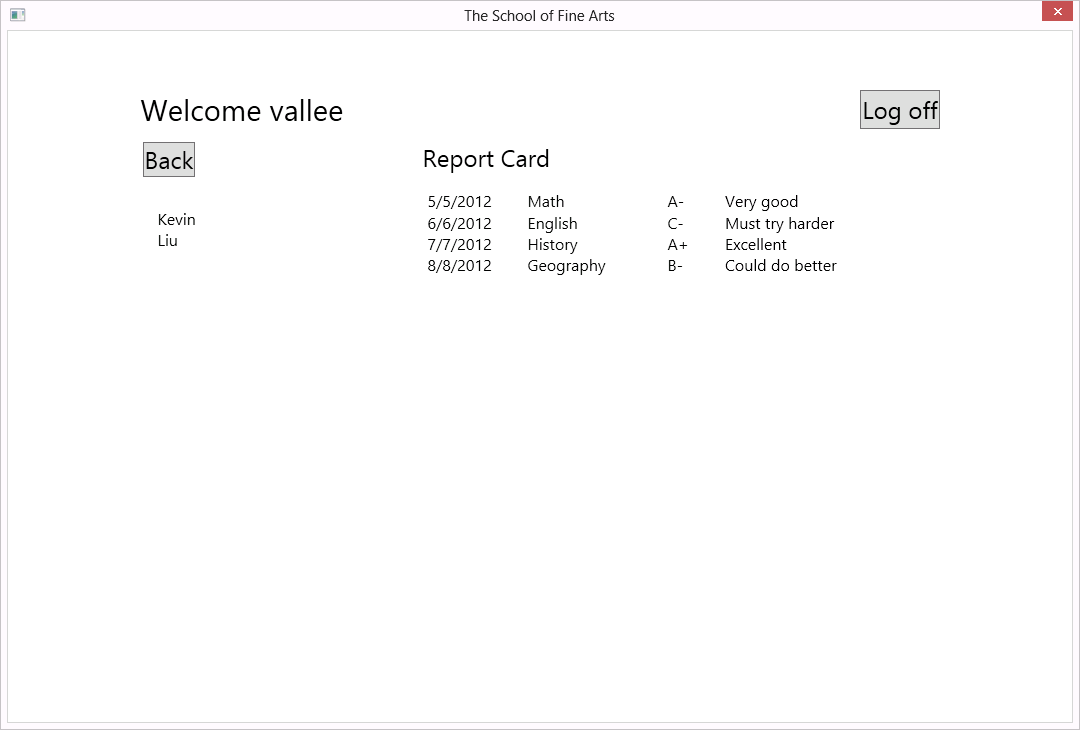
to look like the following markup:

<y:StudentsPage x:Name="studentsPage"

StudentSelected="studentsPage\_StudentSelected"

Visibility="Collapsed" />

#### Task 6: Build and test the application

1. On the **Build** menu, click **Build Solution**.
2. On the **Debug** menu, click **Start Without Debugging**.
3. When the application loads, in the **Username** text box, type **vallee**, and in the **Password** text box, type **password**.
4. Select the **Teacher** check box, and then click **Log on**.
5. Verify that the application displays the **StudentPage** view.
6. Click the student **Kevin Liu** and verify that the application displays the **StudentProfile** view.
7. Click **Log off**.
8. In the **Username** text box, delete the existing contents, and then type **grubere**.
9. Clear the **Teacher** check box, and then click **Log on**.
10. Verify that the application displays the student profile page for **Eric Gruber**.
11. Close the application.
12. On the **File** menu, click **Close Solution**.

**Result:** After completing this exercise, you should have updated the **Grades Prototype** application to respond to user events and move among the application views appropriately.

### Exercise 2: Creating Data Types to Store User and Grade Information

#### Task 1: Define basic structs for holding Grade, Student, and Teacher information

1. In **Visual Studio**, on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, browse to **E:/Allfiles/Mod03/Labfiles/Exercise 2**, click **GradesPrototype.sln**, and then click **Open**.

>**Note :** If any Security warning dialog box appears, clear **Ask me for every project in this solution** check box and then click **OK**.

1. On the **View** menu, click **Task List**.
2. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1a: Create the Grade struct** task.
3. In the code editor, click in the blank line below the comment, and then type the following code:

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; }

}

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

}

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

}

#### Task 2: Examine the dummy data source used to populate the collections

1. In **Solution Explorer**, expand **GradesPrototype**, expand **Data**, and then double-click **DataSource.cs**
2. In the code editor, expand the region **Sample Data**, and then locate the method **CreateData**.
3. Note how the **Teachers ArrayList** is populated with **Teacher** data, each containing **TeacherID**, **UserName**, **Password**, **FirstName**, **LastName**, and **Class** fields.
4. Note how the **Students ArrayList** is populated with **Student** data, each containing **StudentID**, **UserName**, **Password**, **TeacherID**, **FirstName**, and **LastName** fields.
5. Note how the **Grades ArrayList** is populated with **Grade** data, each containing **StudentID**, **AssessmentDate**, **SubjectName**, **Assessment**, and **Comments** fields.
6. On the **File menu**, click **Close Solution**.

**Result:** After completing this exercise, the application will contain structs for the teacher, student, and grade types.

### Exercise 3: Displaying User and Grade Information

#### Task 1: Add the LogonFailed event

1. In **Visual Studio**, on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, browse to **E:/Allfiles/Mod03/Labfiles/Exercise 3**, click **GradesPrototype.sln**, and then click **Open**. >**Note :** If any Security warning dialog box appears, clear **Ask me for every project in this solution** check box and then click **OK**.
3. In the **Task List** window, double-click the **TODO: Exercise 3: Task 1a: Define LogonFailed event** task.
4. In the code editor, click in the blank line below the comment, and then type the following code:

public event EventHandler LogonFailed;

1. In the **Task List** window, double-click the **TODO: Exercise 3: Task 1b: Validate the username and password against the Users collection in the MainWindow window** task.
2. In the code editor, in the **Logon\_Click** method, click in the blank line, and then type the following code:

// 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;
* }
* // If the user is not a teacher, check whether the username and password
* // match those of a student
* else {
* var student = (from Student s in DataSource.Students
* where String.Compare(s.UserName, username.Text) == 0
* && String.Compare(s.Password, password.Password) == 0
* select s).FirstOrDefault();
* // If the UserName of the user retrieved by using LINQ is non-empty
* // then the user is a student
* if (!String.IsNullOrEmpty(student.UserName)) {  
   // Save the details of the student in the global context
* SessionContext.UserID = student.StudentID;
* SessionContext.UserRole = Role.Student;
* SessionContext.UserName = student.UserName;
* SessionContext.CurrentStudent = student;
* // Raise the LogonSuccess event and finish
* LogonSuccess(this, null);
* return;
* }
* }
* // If the credentials do not match those for a Teacher or for a Student
* // then they must be invalid
* // Raise the LogonFailed event
* LogonFailed(this, null);

#### Task 2: Add the Logon\_Failed event handler

1. In the **Task List** window, double-click the **TODO: Exercise 3: Task 2a: Handle logon failure** task.
2. In the code editor, click in the blank line below the comments, and then type the following code:

private void Logon\_Failed(object sender, EventArgs e){

// Display an error message. The user must try again

MessageBox.Show("Invalid Username or Password", "Logon Failed",

MessageBoxButton.OK, MessageBoxImage.Error);

}

1. In **Solution Explorer**, double-click **MainWindow.xaml**.
2. In the XAML editor, locate the task **TODO: Exercise 3: Task 2b: Connect the LogonFailed event of the logonPage view to the Logon\_Failed method in MainWindow.xaml.cs** task.
3. In the line below the comment, modify the XAML markup

<y:LogonPage x:Name="logonPage" LogonSuccess="Logon\_Success"

Visibility="Collapsed" />

to look like the following markup:

<y:LogonPage x:Name="logonPage" LogonSuccess="Logon\_Success"

LogonFailed="Logon\_Failed" Visibility="Collapsed" />

1. In the **Task List** window, double-click the **TODO: Exercise 3: Task 2c: Display the student name in the banner at the top of the page** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

// Display the student name in the banner at the top of the page

txtName.Text = string.Format("Welcome {0} {1}",

SessionContext.CurrentStudent.FirstName,

SessionContext.CurrentStudent.LastName);

1. In the **Task List** window, double-click the **TODO: Exercise 3: Task 2d: Display the teacher name in the banner at the top of the page** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

// Display the teacher name in the banner at the top of the page

txtName.Text = string.Format("Welcome {0} {1}",

SessionContext.CurrentTeacher.FirstName,

SessionContext.CurrentTeacher.LastName);

#### Task 3: Display the students for the current teacher

1. In **Solution Explorer**, expand **Views**, and then double-click **StudentsPage.xaml**.
2. In the XAML editor, locate the **ItemsControl** named **list** and note how data binding is used to display the name of each student. >**Note:** DataBinding is also used to retrieve the StudentID of a student. This binding is used when a user clicks on a student on the **Student Page** list to identify the student whose data is to be displayed in the **Student Profile** page.
3. In the **Task List** window, double-click the **TODO: Exercise 3: Task 3a: Display students for the current teacher (held in SessionContext.CurrentTeacher)** task.
4. In the code editor, in the **Refresh** method, click in the blank line, and then type the following code:

// Find students for the current teacher

ArrayList students = new ArrayList();

* foreach (Student student in DataSource.Students) {
* if (student.TeacherID == SessionContext.CurrentTeacher.TeacherID) {
* students.Add(student);
* }
* }
* // Bind the collection to the list item template
* list.ItemsSource = students;
* // Display the class name
* txtClass.Text = String.Format(“Class {0}”, SessionContext.CurrentTeacher.Class);

1. In the **Task List** window, double-click the **TODO: Exercise 3: Task 3b: If the user clicks on a student, display the details for that student** task.
2. In the code editor, in the **Student\_Click** method, click in the blank line, and then type the following code:

Button itemClicked = sender as Button;

* if (itemClicked != null) {
* // Find out which student was clicked
* int studentID = (int)itemClicked.Tag;
* if (StudentSelected !=null)  
   {  
   // Find the details of the student by examining the DataContext
* // of the Button  
   Student student = (Student)itemClicked.DataContext;  
   // Raise the StudentSelected event (handled by MainWindow to display
* // the details for this student  
   StudentSelected(sender, new StudentEventArgs(student));  
   }
* }

1. In the **Task List** window, double-click the **TODO: Exercise 3: Task 3c: Set the current student in the global context to the student specified in the StudentEventArgs parameter** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

SessionContext.CurrentStudent = e.Child;

#### Task 4: Set the DataContext for the page

1. In the **Task List** window, double-click the **TODO: Exercise 3: Task 4a: Display the details for the current student (held in SessionContext.CurrentStudent)** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

// Bind the studentName StackPanel to display the details of the student

// in the TextBlocks in this panel

studentName.DataContext = SessionContext.CurrentStudent;

* // 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;
* }

1. In **Solution Explorer**, expand **Views** and then double-click **StudentProfile.xaml**.
2. In the XAML editor, locate the task **TODO: Exercise 3: Task 4b: Bind the firstName TextBlock to the FirstName property of the DataContext for this control** task.
3. In the line below the comment, modify the XAML markup

<TextBlock x:Name="firstName" FontSize="16" />

to look like the following markup:

<TextBlock x:Name="firstName" Text="{Binding FirstName}" FontSize="16" />

1. In the XAML editor, locate the task **TODO: Exercise 3: Task 4c: Bind the lastName TextBlock to the LastName property of the DataContext for this control** task.
2. In the line below the comment, modify the XAML markup

<TextBlock x:Name="lastName" FontSize="16"/>

to look like the following markup:

<TextBlock x:Name="lastName" Text="{Binding LastName}" FontSize="16"/>

1. In the **Task List** window, double-click the **TODO: Exercise 3: Task 4d: Create a list of the grades for the student and display this list on the page** task.
2. In the code editor, click at the end of the comment line, press Enter, and then type the following code:

// 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;

#### Task 5: Build and test the application

1. On the **Build** menu, click **Build Solution**.
2. On the **Debug** menu, click **Start Without Debugging**.
3. When the application loads, in the **Username** text box, type **parkerd**, in the **Password** text box, type **password**, and then click **Log on**.
4. Verify that the **Logon Failed** dialog box appears, and then click **OK**.
5. In the **Username** text box, delete the existing contents, type **vallee**, and then click **Log on**.
6. Verify that the **Students** page appears, displaying a list of students.
7. Click the student **Kevin Liu** and verify that the **Student Profile** page is displayed.
8. Click **Log off**.
9. In the **Username** text box, delete the existing contents, type **grubere**, and then click **Log on**.
10. Verify that the **Student Profile** page for **Eric Gruber** is displayed.
11. Close the application.
12. On the **File** menu, click **Close Solution**.

**Result:** After completing this exercise, only valid users will be able to log on to the application and they will see only data appropriate to their role.