## Lab: Retrieving and Modifying Grade Data Remotely

### Lab Setup

Estimated Time: **60 minutes**

### Preparation Steps

1. Initialize the database:
   * In the **Apps** list, click **File Explorer**.
   * In File Explorer, navigate to the **E:/Allfiles/Mod01/Labfiles/Databases** folder, and then double-click **SetupSchoolGradesDB.cmd.**
   * **>** Note: If a Windows protected your PC dialog appears, click **More info** and then click **Run Anyway**. Close File Explorer.

### Exercise 1: Creating a WCF Data Service for the SchoolGrades Database

#### Task 1: Configure data service in the Grades.Web project

1. Open **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/Mod08/Labfiles/Starter/Exercise 1**, click **GradesPrototype.sln**, and then click **Open**.
4. In **Solution Explorer**, right-click the **GradesPrototype** solution, and then click **Restore NuGet Packages**.

#### Task 2: Specify the GradesDBEntities data context for the data service

1. In **Solution Explorer**, in **Grades.Web** project, expand **Services**, and then double-click **GradesWebDataService.svc**.
2. In the code editor, type the following code under all the **using**:

using Grades.DataModel;

1. In **Grades.Web** project, right click **References** then **Add Reference**.
2. Expand **Projects** and in **Solution** select **Grades.DataModel**.
3. Click **OK**.
4. On the **View** menu, click **Task List**.
5. In the **Task List** window, double-click the **TODO: Exercise 1: Task 2a: Replace the object keyword with your data source class name** task.
6. In the code editor, locate the comment **TODO: Exercise 1: Task 2a: Replace the object keyword with your data source class name**, and then type the following code instead of the **object** keyword:

SchoolGradesDBEntities

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 2b: set rules to indicate which entity sets and service operations are visible, updatable, etc** task.
2. In the code editor, click at the end of the comment line, press Enter, and then type the following code:

// Configure all entity sets to permit read and write access.

config.SetEntitySetAccessRule("Grades", EntitySetRights.All);

config.SetEntitySetAccessRule("Teachers", EntitySetRights.All);

config.SetEntitySetAccessRule("Students", EntitySetRights.All);

config.SetEntitySetAccessRule("Subjects", EntitySetRights.All);

config.SetEntitySetAccessRule("Users", EntitySetRights.All);

#### Task 3: Add an operation to retrieve all of the students in a specified class

1. Click after the closing braces for the **InitializeService** method, press Enter twice, and then type the following code:

// Find all students in a specified class.

[WebGet]

public IEnumerable<Student> StudentsInClass(string className)

{

var students = from Student s in this.CurrentDataSource.Students

where String.Equals(s.Teacher.Class, className)

select s;

* return students;
* }

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 2b: set rules to indicate which entity sets and service operations are visible, updatable, etc**. task.
2. In the code editor, click at the end of the comment line, press Enter, and then type the following code:

// Configure the StudentsInClass operation as read-only.

config.SetServiceOperationAccessRule("StudentsInClass",

ServiceOperationRights.AllRead);

#### Task 4: Build and test the data service

1. On the **Build** menu, click **Build Solution**.
2. In **Solution Explore**r, in the **Grades.Web** project, in the **Services** folder, right-click **GradesWebDataService.svc**, and then click **View in Browser (Microsoft Edge)**.
3. In Microsoft Edge, if the **Intranet settings are turned off by default** message appears, click **Don’t show this message again**.
4. Verify that Microsoft Edge displays an XML description of the entities that Data Service exposes.
5. Close the Browser.
6. In Visual Studio, on the **File** menu, click **Close Solution**.

**Result:** After completing this exercise, you should have configured a WCF Data Service in the application to provide remote access to the **SchoolGrades** database.

### Exercise 2: Integrating the Data Service into the Application

#### Task 1: Add an OData Connected Service for the WCF Data Service to the GradesPrototype application

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/Mod08/Labfiles/Starter/Exercise 2**, click **GradesPrototype.sln**, and then click **Open**.
3. In **Solution Explorer**, right-click the **GradesPrototype** solution, and then click **Set StartUp Projects**.
4. In the **Solution ‘GradesPrototype’ Property Pages** dialog box, click **Multiple startup projects**.
5. In the **Action** column for **Grades.Web** and **GradesPrototype**, click**Start**, and then click **OK**.
6. On the **Build** menu, click **Rebuild Solution**.
7. In Solution Explorer, in the **Grades.Web** project, in the **Services** folder, right-click **GradesWebDataService.svc**, and then click **View in Browser (Microsoft Edge)**.

**Note**: Once you have seen that the page opens correctly, close the browser.

1. In **Solution Explorer**, expand **GradesPrototype**, expand **References**, right-click **Grades.DataModel,** and then click **Remove**.
2. Right-click **References**, and then click **Add Connected Service**.
3. In the **Connected Services** window, choose **OData Connected Service**.
4. In the **Configure endpoint** dialog box, in the **Service name** text box, type **Grades.DataModel**.
5. In the **Address** text box, type **http://localhost:1655/Services/GradesWebDataService.svc/$metadata**, and then click **Next**.
6. Click **Finish** and wait until the process completes.
7. If a web page appears discussing the deprecation of WCF Data Services, close the page.
8. In **Solution Explorer**, in the **GradesPrototype** project, expand **Connected Services**, expand the **Grades.DataModel** folder, and then double-click **Reference.cs**.
9. In the code editor, modify the **namespace SchoolGradesDBModel** code to look like the following code:

* namespace Grades.DataModel

1. In **Solution Explorer**, right-click the **GradesPrototype** project, point to **Add**, and then click **New Folder**.
2. Delete the existing name, type **DataModel**, and then press Enter.
3. In **Solution Explorer**, expand the **Grades.DataModel** project, right-click **Classes.cs**, and then click **Copy**.
4. In **GradesPrototype**, right-click **DataModel**, and then click **Paste**.
5. In **Grades.DataModel**, right-click **customGrade.cs**, and then click **Copy**.
6. In **GradesPrototype**, right-click **DataModel**, and then click **Paste**.
7. In **Grades.DataModel**, right-click **customTeacher.cs**, and then click **Copy**.
8. In **GradesPrototype**, right-click **DataModel**, and then click **Paste**.

#### Task 2: Modify the code that accesses the EDM to use the WCF Data Service

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2a: Specify the URL of the GradesWebDataService** task.
2. In the code editor, below the comment, click inside the parentheses, and then type the following code:

new Uri("http://localhost:1655/Services/GradesWebDataService.svc")

1. Add the following code to the **SessionContext** class, after the **Save** method:

static SessionContext()

{

DBContext.MergeOption = System.Data.Services.Client.MergeOption.PreserveChanges;

}

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2b: Load User and Grades data with Students** task.
2. In the code editor, at the end of the comment, press Enter, and then type the following code:

SessionContext.DBContext.LoadProperty(student, "User");

SessionContext.DBContext.LoadProperty(student, "Grades");

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2c: Load User and Students data with Teachers** task.
2. In the code editor, below the comment, click at the end of the **SessionContext.DBContext.Teachers** code, and then type the following code:

.Expand("User, Students")

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2d: Load User and Grades data with Students** task.
2. In the code editor, below the comment, click at the end of the **SessionContext.DBContext.Students** code, and then type the following code:

.Expand("User, Grades")

1. In **Solution Explorer**, in the **GradesPrototype** project, expand **DataModel**, and then double-click **customTeacher.cs**.
2. Click at the end of the **using System.Threading.Tasks** code, press Enter, and then type the following code:

* using GradesPrototype.Services;

1. In the code editor, locate the **TODO: Exercise 2: Task 2e: Refer to the Students collection in the SessionContext.DBContext object** comment in the **customTeacher.cs** file. There are two comments with this text. This is because the comment is located in the **customTeacher.cs** file that you copied from the **Grades.DataModel** project. Make sure that you modify the **customTeacher.cs** file in **DataModel** folder of the **GradesPrototype** project.
2. In the line below the comment, delete the word **Students**, and then type the following code:

* SessionContext.DBContext.Students

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2f: Reference the SessionContext.DBContext.Students collection**.
2. In the code editor, below the comment, change

* SessionContext.DBContext.Students.Local
* to the following code:
* SessionContext.DBContext.Students.Expand("User, Grades")

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2g: Use the AddToGrades method to add a new grade.**
2. In the code editor, below the comment, change

* SessionContext.DBContext.Grades.Add(newGrade);
* to the following code:
* SessionContext.DBContext.AddToGrades(newGrade);

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2h: Load Subject data with Grades** task.
2. In the code editor, below the comment, change **SessionContext.DBContext.Grades** to the following code:

* SessionContext.DBContext.Grades.Expand("Subject")

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2i: Use the AddToStudents method to add a new student** task.
2. In the code editor, below the comment, change

* SessionContext.DBContext.Students.Add(newStudent);
* to the following code:
* SessionContext.DBContext.AddToStudents(newStudent);

#### Task 3: Modify the code that saves changes back to the database to use the WCF Data Service

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 3a: Specify that the selected student has been changed** task.
2. In the code editor, click in the blank space below the comment, and then type the following code:

SessionContext.DBContext.UpdateObject(student);

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 3b: Specify that the current student has been changed** task.
2. In the code editor, click in the blank space below the comment, and then type the following code:

SessionContext.DBContext.UpdateObject(SessionContext.CurrentStudent);

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 3c: Specify that the current user has been changed** task.
2. Click in the blank space below the comment, and then type the following code:

SessionContext.DBContext.UpdateObject(currentUser);

#### Task 4: Build and test the application to verify that the application still functions correctly

1. In **the Solution Explorer**, right click on the **Grades.Prototype** project and select **Set as Startup Project**.
2. On the **Build** menu, click **Build Solution**.
3. On the **Debug** menu, click **Start Without Debugging**.
4. In the **Username** text box, type **vallee**.
5. In the **Password** text box, type **password99**, and then click **Log on**.
6. In the student list, click **Eric Gruber**, and then click **Remove Student**.
7. In the **Confirm** dialog box, click **Yes**.
8. Verify that **Eric Gruber** is removed from the student list.
9. Click **Enroll Student**.
10. In the **Assign Student** dialog box, click **Jon Orton**.
11. In the **Confirm** dialog box, click **Yes**.
12. In the **Assign Student** dialog box, click **Close**, and then verify that **Jon Orton** is added to the student list.
13. Click **Change Password**.
14. In the **Change Password Dialog** dialog box, in the **Old Password** text box, type **password99**.
15. In the **New Password** text box, type **password88**.
16. In the **Confirm** text box, type **password88**, and then click **OK**.
17. In the **Password** dialog box, click **OK**, and then click **Log off**.
18. Click **Log on**, verify that the **Logon Failed** dialog box appears, and then click **OK**.
19. In the **Password** text box, type **password88**, and then click **Log on**.
20. Verify that the student list is displayed.
21. Click **Log off**, and then in the **Username** text box, type **grubere**.
22. In the **Password** text box, type **password**, and then click **Log on**.
23. Verify that the student profile for Eric Gruber appears, and then click **Log off**.
24. Close the application.
25. In Visual Studio, on the **File** menu, click **Close Solution**.

**Result:** After completing this exercise, you should have updated the **Grades Prototype** application to use OData Connected Service.

### Exercise 3: Retrieving Student Photographs Over the Web (If Time Permits)

#### Task 1: Create the ImageNameConverter value converter class

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/Mod08/Labfiles/Starter/Exercise 3**, click **GradesPrototype.sln**, and then click **Open**.
3. In **Solution Explorer**, right-click the **GradesPrototype** solution, and then click **Set StartUp Projects**.
4. In the **Solution ‘GradesPrototype’ Property Pages** dialog box, click **Multiple startup projects**.
5. In the **Action** column for **Grades.Web** and **GradesPrototype**, click **Start**, and then click **OK**.
6. On the **Build** menu, click **Rebuild Solution**.
7. In the **Task List** window, double-click the **TODO: Exercise 3: Task 1: Create the ImageNameConverter value converter to convert the image name of a student photograph into the URL of the image on the Web server** task.
8. Click at the end of the **// Converter class for transforming an image name for a photograph into a URL** comment, press Enter, and then type the following code:

public class ImageNameConverter : IValueConverter { }

1. In the **ImageNameConverter** class, type the following code:

const string webFolder = "http://localhost:1650/Images/Portraits/";

1. Right-click the **IValueConverter** keyword, point to **Quick Actions and refactorings**, and then click **Implement Interface**.
2. In the **Convert** method, delete the existing statement that throws a **NotImplementedException**, and then type the following code:

* string fileName = value as string;  
    
  if (fileName != null)  
  {  
   return string.Format("{0}{1}", webFolder, fileName);  
  }  
  else  
  {  
   return string.Empty;  
  }

1. On the **Build** menu, click **Build Solution**.

#### Task 2: Add an Image control to the StudentsPage view and bind it to the ImageName property

1. In **Solution Explorer**, in the **GradesPrototype** project, expand **Views**, and then double-click **StudentsPage.xaml**.
2. In the XAML editor, in the **UserControl** element at the top of the markup, click after the **xmlns:d=“http://schemas.microsoft.com/expression/blend/2008”** line, press Enter, and then type the following markup:

xmlns:local="clr-namespace:GradesPrototype.Views"

1. Locate the **TODO: Exercise 3: Task 2a. Add an instance of the ImageNameConverter class as a resource to the view** comment, click at the end of the comment, press Enter, and then type the following markup:

<UserControl.Resources>

<local:ImageNameConverter x:Key="ImageNameConverter"/>

</UserControl.Resources>

1. Locate the **TODO: Exercise 3: Task 2b. Add an Image control to display the photo of the student** comment, click at the end of the comment, press Enter, and then type the following markup:

<Image Height="100" />

1. Locate the **Exercise 3: Task 2c. Bind the Image control to the ImageName property and use the ImageNameConverter to convert the image name into a URL** comment.
2. In the line above the comment, modify the xml

<Image Height="100" />

markup to look like the following markup:

<Image Height="100" Source="{Binding ImageName,

Converter={StaticResource ImageNameConverter}}" />

#### Task 3: Add an Image control to the StudentProfile view and bind it to the ImageName property

1. In Solution Explorer, double-click **StudentProfile.xaml**.
2. Locate the **TODO: Exercise 3: Task 3a. Add an instance of the ImageNameConverter class as a resource to the view** comment.
3. Click at the end of the comment, press Enter, and then type the following markup:

<app:ImageNameConverter x:Key="ImageNameConverter"/>

1. Locate the **TODO: Exercise 3: Task 3b. Add an Image control to display the photo of the student and bind the Image control to the ImageName property and use the ImageNameConverter to convert the image name into a URL** comment.
2. Click at the end of the comment, press Enter, and then type the following markup:

<Image Height="150" Source="{Binding ImageName,

Converter={StaticResource ImageNameConverter}}" />

#### Task 4: Add an Image control to the AssignStudentDialog control and bind it to the ImageName property

1. In **Solution Explorer**, expand **Controls**, and then double-click **AssignStudentDialog.xaml**.
2. In the XAML editor, at the top of the markup, click at the end of the **xmlns:x=“http://schemas.microsoft.com/winfx/2006/xaml”** line, press Enter, and then type the following markup:

xmlns:local="clr-namespace:GradesPrototype.Views"

1. Locate the **TODO: Exercise 3: Task 4a. Add an instance of the ImageNameConverter class as a resource to the view** comment.
2. Click at the end of the comment, press Enter, and then type the following markup:

<Window.Resources>

<local:ImageNameConverter x:Key="ImageNameConverter"/>

</Window.Resources>

1. Locate the **TODO: Exercise 3: Task 4b. Add an Image control to display the photo of the student and bind the control to the ImageName property and use the ImageNameConverter to convert the image name into a URL** comment.
2. Click at the end of the comment, press Enter, and then type the following markup:

<Image Height="100" Source="{Binding ImageName,

Converter = {StaticResource ImageNameConverter}}" />

#### Task 5: Build and test the application, verifying that student’s photographs appear in the list of students for the teacher

1. On the **Build** menu, click **Build Solution**.
2. On the **Debug** menu, click **Start Without Debugging**.
3. In the **Username** text box, type **vallee**.
4. In the **Password** text box, type **password88**, and then click **Log on**.
5. Verify that the student list now includes images.
6. \Click **George Li**, and then verify that the student profile appears with an image.
7. Click **Remove Student**.
8. In the **Confirm** dialog box, click **Yes**.
9. Click **Enroll Student**.
10. In the **Assign Student** dialog box, verify that each of the unassigned students has an image.
11. Click **George Li**.
12. In the **Confirm** dialog box, click **Yes**.
13. In the **Assign Student** dialog box, click **Close**.
14. Verify that George Li is added to the student list with an image.
15. Close the application.
16. On the **File** menu, click **Close Solution**.

**Result:** After completing this exercise, the student list, student profile, and unassigned student dialog box will display the images of students that were retrieved across the web.