Programming 3 Assignment 5

Assignment Overview

- The specifications associated with this assignment will will introduce students to the use of WCF Web Services. The WCF Service will be used to process Registration and Grading business processes.
- Work associated with this assignment will reinforce the student's learning in the following areas:
    - Development and use of WCF Web Services.

Evaluation

- The evaluation of this assignment will be in the form of a Code Review and Runtime Demonstration.
- During the Code Review evaluation will be based on:
    - Comments
    - Documentation
    - Adherence to Standards – and Pattern Best Practices
    - Coding choices (i.e. refactoring, no repeated code, readability etc…)
    - Implementation of Requirements
    - Verbal discussion of code
- During the Run-Time Demonstration evaluation will be based on:
    - A variety of run-time tests

Specifications

WCF Service project

- Add a new **WCF Service Application** project to the existing solution. Name this project: **BITCollegeService**.
- Within the WCF Service, add a reference to the BITCollege_FL MVC project
- Create a Reference to the Utility project – **Note**: The explicit reference to Utility may not be needed as it may be added implicitly when the BITCollege_FL reference is created.
- **NOTE**: NO SERVICE REFERENCES WILL BE CREATED DURING THIS ASSIGNMENT. SERVICE REFERENCES WILL BE CREATED IN ASSIGNMENTS 6 & 7 WHEN THE WCF SERVICE PROJECT IS UTILIZED.

Programming 3 Assignment 5

Entity Framework

- By default WCF Service projects do not include Entity Framework. Use Nuget to install Entity Framework to this project. See Module 05 Nuget Packages in the Course Notes for instructions.

Database Connection

- Copy the ***<connectionStrings>*** block (including the opening and closing tags) from the Web.Config file of the BITCollege_FL MVC project to the Web.Config file of the WCF Web Service project.
- Place the copied ***<connectionStrings>*** block just below the closing ***</configSections>*** tag.

Service Object

- Remove the following files from your WCF Service project – as they do not have meaningful names:
    - IService1.cs
    - Service1.svc
    - Service1.svc.cs (as a child of Service1.svc)
- Define a WCF Service object called CollegeRegistration which will generate three files:
    - CollegeRegistration.svc
    - ICollegeRegistration.cs
    - CollegeRegistration.svc.cs (as a child of CollegeRegistration.svc)

ICollegeRegistration

- Further define ICollegeRegistration.cs by creating Operation Contracts based on the Interface shown below:

Programming 3 Assignment 5

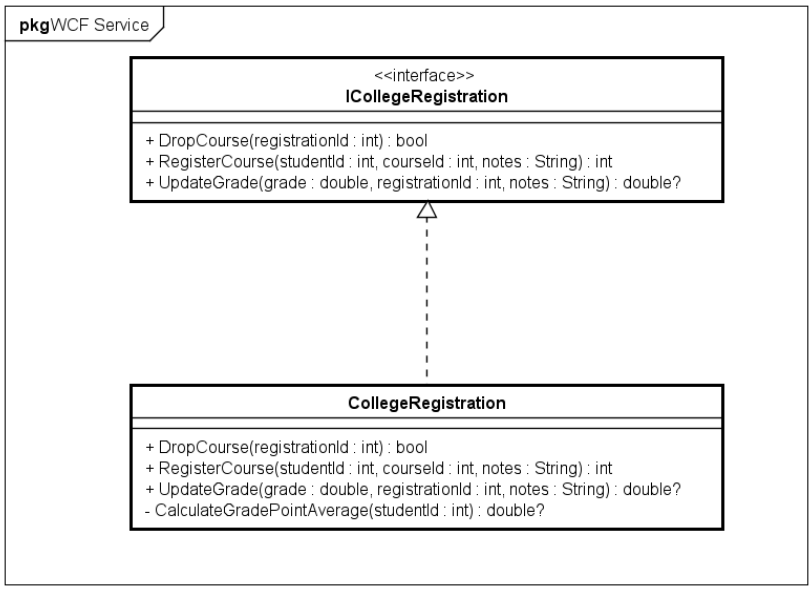


Figure 1

CollegeRegistration

- Implement the operation contracts in CollegeRegistration.cs as follows:

Instance Variables (Private Fields)

- Instantiate an object of the BITCollege_FLContext class as defined in the Data folder of the BITCollege_FL project.
- Define additional instance variable(s) as necessary to accomplish the requirements that follow.

Private Methods/Refactoring

- For each of the methods that follow there may be opportunity to refactor such that duplicate code is reduced/eliminated.
- Define private method(s) as necessary to reduce/eliminate repetitive code.

DropCourse

- Use *LINQ-SQL-Server* (Lambda or Traditional Syntax) to retrieve the Registration record from the database which corresponds to the method argument.
- Remove the record from the database and persist this change.

- If an exception occurs, return a value of false, otherwise return a value of true.

RegisterCourse

Discussion

- This method will make use of various return codes to indicate success or failure. In the event that the course registration fails, the return code will indicate the reason.
- A registration **will not** be permitted under the following circumstances. Use *LINQ-SQL-Server* (Lambda or Traditional Syntax) to validate the following:
    - When the student already has an incomplete registration for the same course. This situation will result in a return code of -100.
    - When a student is registering for a Mastery course and this registration would exceed the maximum attempts value for the Mastery course. This situation will result in a return code of -200.

Implementation

- To perform the appropriate validation on potential registrations, previous registrations must be examined.
    - Define an IQueryable<Registration> query retrieving all records from the Registrations table involving the Student and Course represented by method parameter values.
- Before registering for the course, information about the course must be known.
    - Define a query to retrieve the Course record represented by the corresponding method parameter value.
- If the registration is possible, the Student record must be updated such that their outstanding fees reflect the cost of the registration.
    - Define a query to retrieve the Student record represented by the corresponding method parameter value.

Incomplete Registration Check

- Using the IQueryable<Registration> query completed above, further query to determine if any of the Registration records involving the Student and Course represented by method parameter values has a grade of null.
    - If so, the student is not permitted to register for this course because they already have an ungraded registration.
    - Set the return code value to -100.

Programming 3 Assignment 5

Mastery Attempt Check

- Using the Course query completed above, determine whether the course is a Mastery course (**Hint**: Check the CourseType property using a helper method included with the BusinessRules class).
    - If so, cast the Course query above to be of MasteryCourse type.
    - Obtain the MaximumAttempts value.
- Using the IQueryable<Registration> query completed above, determine the number of registrations that have already taken place between the Student and Course in question.
- If the number of registrations is greater than or equal to the MaximumAttempts value, then the upcoming registration would exceed the MaximumAttempts allowed.
    - If so, set the return code value to -200.

Register for Course

- If the above validation indicates that the registration can proceed:
    - Create a new Registration object
    - Populate this object's properties with values pertinent to this registration:
        - For the StudentId, CourseId and Notes fields, use the method arguments.
        - For the RegistrationDate, use today's date.
        - Use predefined functionality to generate the RegistrationNumber.
        - Add the Registration object to the Registrations table.
        - Persist this new record to the database.
- The student must now be charged through their OutstandingFees for the new Registration.
    - Using the Course query above, determine the TuitionAmount of the Course.
    - Update the Student record by adding the Adjusted TuitionAmount to the OutstandingFees property.
        - Ensure that the student is charged the appropriate fees based on the TuitionRateAdjustment method of the Student's GradePointState.
    - Persist this change to the database.
- If the above code results in an exception, a return code of -300 will be used.

- Ensure the appropriate return code is returned from this routine
    - If the registration is successful, return a value of 0.
    - If an exception occurs while updating, return a value of -300.
    - If the student has exceeded the MaximumAttempts of a Mastery course, return a value of -200.
    - If the student already has an ungraded registration for this course, return a value of -100.

- **Note**: Methods should have only one exit. So ensure that only one return statement is used when coding this method.

UpdateGrade

- Use *LINQ-SQL-Server* (Lambda or Traditional Syntax) to retrieve the Registration record from the database which corresponds to the method argument.
- Set the Grade property of the Registration record to the value of the grade argument.
- Modify the Notes property with the value of the method argument.
- Persist the updated grade to the database.
- Call the CalculateGradePointAverage method (below) passing the appropriate argument.
  - Capture the result of the CalculateGradePointAverage method into a local variable.
- Return the result of the CalculateGradePointAverage method.

Programming 3 Assignment 5

CalculateGradePointAverage

Discussion

- A number of variables will be needed to complete this calculation. To simplify this process the following list of variables has been provided along with their descriptions. You are encouraged to use these variables when coding this method:
    - double grade: This variable will store the grade of an individual registration.
    - CourseType courseType: This variable will store the type of course that the individual registration represents (e.g. CourseType.MASTERY, CourseType.AUDIT, CourseType.GRADED).
    - double gradePoint: This variable will store the grade point value for an individual registration.
    - double gradePointValue: This variable will store the calculated grade point value for an individual registration based on the course credit hours.
    - double totalCreditHours: This variable will store the accumulated credit hours value of ALL registrations.
    - double totalGradePointValue: This variable will store the accumulated grade point value for ALL registrations.
    - Double? calcuatedGradePointAverage: This variable will store the calculated grade point average and will be returned from the method.

Implementation

- Use *LINQ-SQL-Server* (Lambda or Traditional Syntax) to define an IQueryable<Registration> result set containing all Registration records with a Grade value that is not null which belong to the student represented by the method argument.
- Iterate through this result set
    - **Note**: when coding your foreach loop, the collection of Registration records will need to have a .toList() method appended to the collection.
        - E.g. `foreach (Registration record in registrations.ToList())`
    - Obtain the Grade for the registration
    - Determine the Course Type for the registration (E.g. Mastery, Graded or Audit)
        - Obtain the corresponding CourseType enum value
            - **Hint**: Use a given method from the Utility project
    - Exclude any Audit courses from the GPA calculation
    - Use given functionality in the Utility project's BusinessRules class to:
        - Use the Course Type and grade to determine the corresponding Grade Point Value for the grade.

- The Grade Point Average formula is as follows:
  - Multiply each registration's GradePointValue by the Course's CreditHours.
    - **Hint**: The Course's credit hours can be obtained through a navigation property.
      - E.g. `record.Course.CreditHours`
  - Determine Total Grade Point Value by accumulating the above value for all registrations.
  - Determine the Total Credit Hours by accumulating the CreditHours for all registration.

- If the Total Credit Hours value is 0 following the above iterations:
  - Set the calcuatedGradePointAverage variable to NULL
- Otherwise
  - Set the calcuatedGradePointAverage variable to the result of dividing the Total Grade Point Value by the Total Credit Hours

- Define a LINQ Query (Traditional or Lambda syntax) to obtain the Student record to which the newly calculated Grade Point Average applies.
- Set the GradePointAverage property of the Student record to the newly calculated Grade Point Average.
- Persist this change.
- Ensure that any changes to the Student's GradePointAverage cause the student to be placed in the appropriate GradePointState.
- Return the calculated Grade Point Average.

WCF Service Project

- Build the WCF Web Service.

Utility Project: BusinessRules

- Update the BusinessRules class by completing the following static method:

  + RegisterError(errorCode : int) : String

- Implement the method as follows:
- Evaluate the errorCode and return the corresponding string based on the following:

Error Code	Return Value
-100	***Student cannot register for a course in which there is already an ungraded registration.***
-200	***Student has exceeded maximum attempts on mastery course.***
-300	***An error has occurred while updating the registration.***
Any other value	***Unknown error.***

Test

- Use the WCF Test Client to test each of the methods of the WCF Service.
- Feel free to modify the data in your database to match one or more of the scenarios outlined in the WCF Service so that you can test your results.

Examples GPA Calculations

Given the following grades:

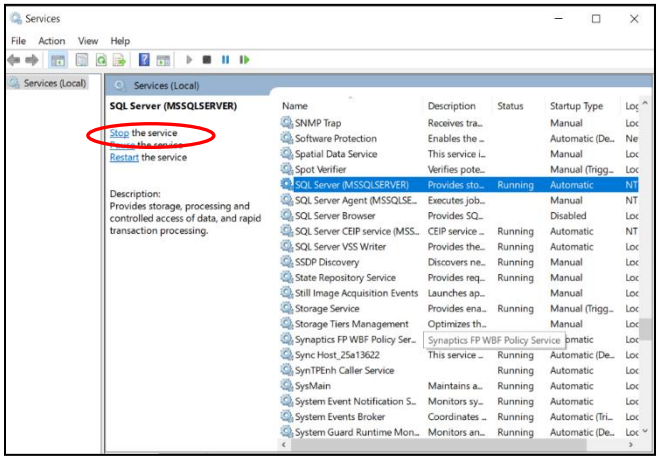
Numeric Grade	Grade Point Value	Credit Hours	Calculated GPA
.34	0	5	((0 * 5) + (3.5 * 5) + (2.5 * 5)) / 15 = **2.00**
.75	3.5	5	
.65	2.5	5	

Numeric Grade	Grade Point Value	Credit Hours	Calculated GPA
.89	4	2	((4 * 2) + (3 * 5) + (3.5 * 2)) / 9 = **3.333**
.72	3	5	
.77	3.5	2	

Backup

- Backup your Solution.
- Backup your database and its corresponding log file.
- The database and log file can be found in the following directory:
- C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA

- **Note**: it may be necessary to stop the SQL Server Service to perform this backup.



Prior to Evaluation

- Specific (sanitized) test data will be used for the evaluation of this assignment.
- In the Assignment 05 Files/SQL Server Script folder you will find a sql server script. Use this script in the instructions that follow:

1. Ensure you have created a backup of your database.

2. Modify the script called ***A5TestData.sql*** found in the Assignment 5 Files\SQL Server Scripts folder by replacing the [DATABASENAME] placeholder with the name of your BITCollege_FLContext database.

3. Run the script called ***A5TestData.sql*** found in the Assignment 5 Files\SQL Server Scripts folder.

   Two records should be inserted into the Students table starting with the primary key value of 501.

StudentId	GradePointStateId	AcademicProgramId	StudentNumber	FirstName	LastName	Address	City	Province	DateCreated	GradePointAverage	OutstandingFees	Notes
501	4	2	50000001	Assignment 5	Student 1	501 Assignment 5 Road	Winnipeg	MB	2020-01-01 00:00:00.000	4.5	500	Expected Key 501
502	2	2	50000002	Assignment 5	Student 2	502 Assignment 5 Road	Winnipeg	MB	2020-01-01 00:00:00.000	2.44	12.5	Expected Key 502

Nine records should be inserted into the Courses table starting with the primary key value of 501.

CourseId	AcademicProgramId	CourseNumber	Title	CreditHours	TuitionAmount	Notes	AssignmentWeight	ExamWeight	MaximumAttempts	Discriminator
501	4	G-500001	A5 Graded 1	5	1220	Expected Key 501	0.3	0.7	NULL	GradedCourse
502	2	G-500002	A5 Graded 2	3	1220	Expected Key 502	0.1	0.9	NULL	GradedCourse
503	2	G-500003	A5 Graded 3	5	1000	Expected Key 503	0.1	0.9	NULL	GradedCourse
504	2	M-50001	A5 Mastery 1	3	1220	Expected Key 504	NULL	NULL	4	MasteryCourse
505	2	M-50002	A5 Mastery 2	5	1220	Expected Key 505	NULL	NULL	1	MasteryCourse
506	2	M-50003	A5 Mastery 3	5	500	Expected Key 506	NULL	NULL	1	MasteryCourse
507	1	A-5001	A5 Audit 1	2	1220	Expected Key 507	NULL	NULL	NULL	AuditCourse
508	4	A-5002	A5 Audit 2	1	240	Expected Key 508	NULL	NULL	NULL	AuditCourse
509	3	A-5003	A5 Audit 3	2	240	Expected Key 509	NULL	NULL	NULL	AuditCourse

Fourteen records should be inserted into the Registrations table starting with the primary key value of 501.

RegistrationId	StudentId	CourseId	RegistrationNumber	RegistrationDate	Grade	Notes
501	501	505	5501	2022-08-03 00:00:00.000	0.95	Expected Key 501
502	501	501	5502	2022-08-03 00:00:00.000	0.92	Expected Key 502
503	501	502	5503	2022-08-03 00:00:00.000	0.98	Expected Key 503
504	501	506	5504	2022-07-01 00:00:00.000	0.95	Expected Key 504
505	501	504	5505	2022-07-05 00:00:00.000	NULL	Expected Key 505
506	501	503	5506	2022-07-05 00:00:00.000	NULL	Expected Key 506
507	502	505	5507	2022-08-03 00:00:00.000	0.65	Expected Key 507
508	502	501	5508	2022-08-03 00:00:00.000	0.62	Expected Key 508
509	502	502	5509	2022-08-03 00:00:00.000	0.68	Expected Key 509
510	502	506	5510	2022-07-01 00:00:00.000	0.65	Expected Key 510
511	502	504	5511	2022-07-05 00:00:00.000	NULL	Expected Key 511
512	502	503	5512	2022-07-05 00:00:00.000	NULL	Expected Key 512
513	501	508	5513	2022-07-05 00:00:00.000	0.67	Expected Key 513
514	502	509	5514	2022-07-05 00:00:00.000	0.81	Expected Key 514

4. **Note**: If any testing based on the records inserted above takes place prior to the evaluation:
   a. Delete the newly inserted Student and Course records from the database.

```
DELETE FROM [dbo].[Students]
      WHERE StudentId > 500

DELETE FROM [dbo].[Courses]
      WHERE CourseId > 500
```

      i. The above will cause a cascade delete of related Registrations records.
   b. Re-run the ***A5TestData.sql*** script.

Programming 3 Assignment 5

Evaluation: In Person/Hybrid Delivery

- This assignment must be evaluated on or before the due date/time specified in Learn.
    - The assignment is due at the beginning of class on the due date specified. Your instructor will evaluate the assignment on the students' machine in a random order. Remote evaluations will be based on a first come, first served basis.