Computer Science

COMPSCI 280

Assignment 2: C# application with database connection

In this assignment, you will create an ASP .NET Model-View-Controller (MVC) application in C# using Visual Studio (VS) and connect to, read and modify a MySQL database from that application. This will give you hands-on experience working with C#, applying the MVC approach to producing a Web application, and in working with databases under programmatic control.

This assignment is due on **Friday, 26 September 2014, 12 noon**, and is worth 5% of your final course marks.

Estimated time to complete this assignment: 15-20 hours.

Submission

Submit via the Assignment Dropbox (https://adb.auckland.ac.nz/). Create a single .zip file of the entire solution directory and submit that **one file** via the dropbox. Note:

- These MVC applications make a *lot* of files – the solution might be around 525 files in 210 folders and about 64MB uncompressed. Even compressed it'll still be about 23MB, so allow appropriate time for the upload.
- Ensure that you submit for the correct course.
- If you resubmit, please include **all** files in the .zip file of your re-submission.

Requirements

A) Creating the assignment project

1. Create your project with Microsoft Visual Studio (2012 or 2013).
2. Have 'Create directory for solution' ticked and have the Name and Solution name be 'Assignment2' (no spaces, no quotes; the marker will be expecting exactly this name)
3. Create a Visual C# ASP .NET MVC 4 Web Application using the default 'Internet Application' template with the Razor view engine

B) Setting up the database

1. Use your MySQL database from Assignment 1 as the starting point for this assignment.
2. Your MVC application will connect to that database. Note:
    a. If you're working from home, you'll want to work with a copy of the database and you'll use a different connection string (i.e. to' localhost' rather than 130.216.39.235)
    b. When you submit the assignment, however, you *must* have it connect to 130.216.39.235. The marker will mark the application based on how it runs on the version of the database that you have there.
    c. Remember to have your database password be something *other* than your University password or any password important to the rest of your life (e.g. not your Facebook password, not your banking PIN).
3. We're going to be working quite a lot with the `flightcabincrew` table, which represents the assignment of people as cabin crew for particular flights. Add a `startDate` field to that table, representing when a specified crew member starts working a particular flight.

C) Establishing interoperability

1. In order for the MVC application to work with MySQL you'll need to add References in the project for MySql.Web.dll, MySql.Data.Entity.EF5.dll and MySql.Data.dll
    a. These files are available from the Cecil resources for the course (or by installing MySQL Connector Net 6.7.5 from the Web, available from http://dev.mysql.com/downloads/connector/net/6.7.html)
    b. Place the files in the 'bin' folder for your project and add References as per the Week 7 labsheet

D) The main screen

1. The main (index) screen of your MVC application should open to a tabular listing of all the flight cabin crew assignments. This list should have the following columns:
    a. Person name
    b. The flight day (i.e. "Monday" or such)
    c. The "From" airport for the route (show the 3-digit code, e.g. "AKL")
    d. The "To" airport for the route (also as 3-digit code)
    e. The aircraft model for that route (e.g. "A380")

    f. The start date of that person's assignment to crew that route in yyyy-mm-dd format
2. This list should 'page' (i.e. offer only a limited number of rows and links to see more rows: *hint* – the default WebGrid behaviour is fine here)

E) Add a record

1. Include a link on the main screen to another screen where the user can enter a new cabin crew assignment (person, flight and start date).
2. When the user submits the new data, if it is all valid, then the user should be returned to the main screen and the new record should be in the list.
3. Provide data entry support and validation on the data entered by the user
    a. The user should have access to a date picker control for entering the date
    b. The screen should provide a default start date of today
    c. The user should receive feedback if the date is invalid
    d. The crew person should be selected by name using a dropdown list
    e. Flight should be selected from a dropdown list showing each flight with its flight day, "from" airport and "to" airport (indicating airports by 3-digit code)
        - E.g. a valid dropdown list entry might be "Monday AKL WLG"
    f. If the user attempts to add a person that is not qualified to be cabin crew for the aircraft type of that flight, then the user should receive a feedback message, "Crew member not qualified for the specified flight"
        - A crew member is qualified to be cabin crew for a flight if they have a record in the `CabinCrew` table that matches the person by `forAircraftType` to the `aircraftType` of the aircraft used for that flight

F) Delete a record

1. On the list of cabin crew assignments on the main screen, for each record provide a Delete link
2. When the user selects Delete, remove that record from the list

G) Personnel qualifications view

1. Include a link on the main screen labelled "View personnel qualifications" that takes the user to a new screen
2. On that screen show a list with one row for each person. It should have two columns
    a. Person name
    b. The set of aircraft for which that person is qualified to be cabin crew
3. Only include a row in the list for people who are qualified to be cabin crew on at least one type of aircraft
4. If a person is qualified to be cabin crew for multiple types of aircraft then the second column of the table should show a comma-delimited list (e.g. "B787, B777, B767")
5. Include a link on this personnel qualifications screen to allow the user to return to the main screen

H) Look and feel

1. Remove all material from the default MVC template that is inappropriate to the application
    a. E.g. remove the "To learn more about ASP.NET MVC visit…" link from the main screen
    b. Note: you can leave the Register and Login links and don't need to modify their associated screens
2. Modify the following parts of the template to appropriate content for this application (you can use your imagination a bit here)
    a. The 'your logo here' (you can leave it as text or add an image… if you use an image, have it be one that's available from an external URL or given on a relative path so that it still works for the marker)
    b. The About page – this should just briefly describe the present assignment
    c. The Contact page – give your contact info so the marker can reach you if necessary
    d. Copyright (appears at the bottom of the screen) – you can claim copyright for yourself
3. Colour scheme – change it from the default provided. Keep it readable (e.g. *not* high saturation blue text on a red background or such)
4. Label the screens and content appropriately, including the title as it appears on the browser tab, page heading and all table column headings
5. Give your tables appropriate column widths (e.g. headings not rammed right against one another) and formatting (e.g. don't include time component in display of dates)

(Note that the 'look and feel' section is not a beauty contest – you get marks for having done the modifications, but no extra marks for making it extra pretty.)

Hints

You might find it helpful to consider the following:

- For a table that requires the combination of values across multiple fields to be unique (like `FlightCabinCrew`), the MVC model's class definition should specify each field that is part of the composite key using an attribute definition `[Key, Column(Order = 1)]` (and successive 'Order' values for the other fields of the composite key)
- You haven't been shown how to do dropdown lists in lecture or in the labsheets. To define a dropdown list of values from a database table:
    - You can use the ViewBag to communicate the list between the Controller and the View
    - In the Controller, make an assignment like
      `ViewBag.list = new SelectList(p.ToList(),"id","name");`
      `p` should be the result of a query that defines your list's contents; "id" indicates the name of the field from the query result for which the dropdown will return a value (e.g. a Person id or the like); "name" indicates the name of the field that will show to the user on the list (e.g. a Person's name)
    - In the View, include a Razor declaration like:
      `@Html.DropDownListFor(model => model.cabinCrewId, (SelectList)ViewBag.list)`
      (note how we have to cast the `ViewBag.list` item to type `SelectList` – since ViewBag is a dynamic 'Expando' class object, the compiler doesn't know what type its `list` attribute might be)
- For the 'Personnel qualifications' view, note that you are not required to use a WebGrid control. You can; but if you find it easier to create your own HTML table, that's OK too (i.e. you aren't required to provide a paging function for the data on that screen).

Note that your marks accumulate based on what you accomplish. If you don't have time to meet all of the requirements A-H above, do the ones you can and still submit your work on time. If you're just missing one or two things, the penalties are pretty small.

Note that the assignment will be a lot easier if you've done the labsheets.

Mark Scheme

The marker will be looking for the following:

Item	Points
Main screen	
Shows all required fields	2
Fields correctly coded/formatted (airport codes, yyyy-mm-dd date)	2
Provides pagination of record list	1
Add a record	
Allows addition of a new cabin crew assignment, including date	2
Date uses date picker and defaults to today	2
Feedback on invalid date	1
Dropdown list for person, showing names on list	1
Dropdown list for flight, showing day and from-to airports by code	2
Validation against adding person not qualified to work that flight	2
Delete a record	
Link on each row of main screen's table that deletes record	2
Personnel qualifications	
Shows qualifications per person where person qualified on at least one aircraft type	1
Shows multiple qualifications with comma-delimited list of aircraft models	2
Look and feel	
No inappropriate/extraneous material from template	1
Customised logo, About page, Contact page, copyright notice and colour scheme	2
Appropriate labelling in title, page headings and column headings	1
Good table appearance (e.g. column widths, date format)	1
Total	**25**

And remember…

After you've submitted your assignment, don't change your MySQL database at 130.216.39.235 as you don't know when the marker will be accessing it – don't change the tables that are used by your application and don't change the password.

If you did most of your work at home: Before zipping and submitting the project to the dropbox, remember to update the Web.config entry for connecting from the University (as per the week 7 labsheet) and keep in mind that it's the 130.216.39.235 version of the database, not the one you had at home, that the marker will use!