W3D1 -- Let’s Change A Few Things.

So we created a new datatype and manipulated a Database, right? The next thing we should do is change how we get that information and publish it. Lets do that. The first thing we should attempt to do is merge add student and the list pages.

Understanding the “Create Page”

1. So lets look the create page.
   1. You should notice that it contains a form. This form has specific elements in it. The most important one is the one labeled with `asp-for=”…”` This is to put it simply this element of the label and the input fields tells the cs file that accompanies the cshtml file what is in that field, and where to put the data that is input into that field.
   2. Notice the classes “form-control” “text-danger” and “control-label.” These are important formatting elements provided by Bootstrap. We will look into bootstrap later.
   3. Notice the `method=”post”` in the <form> element. This is a the “only” way to get data back to the server. It is the “only” way because the other way, get, posts things to the URL. If you are passing sensitive information like usernames and passwords you can’t use get. Post also allows you to pass more information between the client and the server.
2. Now lest open “create.cshtml.cs”
   1. The first thing you should notice is the name of the class, “CreateModel” this expands “PageModel” this put simply dictates what this page does.
   2. Now look at that `[Bind Property]` section. This is the section that allows you to manipulate the database. Take now of how this works.
      1. It creates a student variable.
      2. Then there is an async function that runs on a “submit” function.
      3. It checks to make sure the model state is valid and if it is not then it returns the page (i.e. sends the user back to the page)
      4. It adds the “student” to the context i.e. the database
      5. It then saves changes and then takes you back to the index page.

Understanding The “Index Page”

1. Lets open Index.cshtml.
   1. It is a table, no surprise.
   2. It sets up the table.
      1. It sets up the <thead> by grabbing the category names for the model.
      2. It then goes into a “foreach” and displays each “Student in the Database.
      3. It displays each item then using an <a> link to link to the other pages.
2. In index.cshtml.cs
   1. This shouldn’t be much different from Create.cshtml.cs. it contains only the first half of the page though. However it does create a list called “student” that then gets populated from the database.

Combining the 2

1. Let’s modify the index page such that we can merge the create page into it.
   1. On index.cshtml.cs
      1. Copy paste the [Bind Property] section in create.cshtml.cs into index.cshtml.cs.
      2. Change student variable in the [bind property] to student2 (there should be 2 locations in Index.cshtml.cs that you should do this.)
      3. Change nothing else.
   2. On Index.cshtml
      1. Copy the form from create.cshtml, into index.cshtml just before the ending tag for table.
      2. Change all the <div> to <td> except the first one.
      3. Add a <tr> before the first <div> and terminate it after the last <td>.
      4. Change all the “student” elements in the section you just pasted to Student2.
      5. Read the errors you get from doing this.
      6. Run the page.
         1. Fix errors if you get any, this will likely be because you didn’t change the student to student2 somewhere. Otherwise ask.
      7. Now go and fix your HTML. Form before Table, and Div after form.
      8. Save and reload, do you notice any differences?

What have you learned?

1. Briefly describe how database interaction works or is handled for you in ASP.net
2. Are there any benefits to coding in such a way you don’t have to worry about database interactions and sending, lets say commands to a sql server?
3. Would you rather code in such a way that you can interact with the database directly?
4. Do you think ASP.Net would be vulnerable to SQL injections, note that MS’s Entity framework already protects against SQL Injection?
5. Try to inject some SQL and see what happens.
6. Go watch this video. <https://channel9.msdn.com/Series/aspnetmonsters/ASPNET-Monsters-105-SQL-Injection-attacks-in-Entity-Framework-Core-20>