Skipping to DB connectivity:

1. Database-First
2. code first

!

DbContext => A bridge between your application and DB.

→ DbSet<T> => represents a collection of entities.

① DbContext class is also made in the Models folder. The name of the class will be the name of the DB.

② Constructor in the DbContext class:

```
public ClassName (Dbcontext@Options option):
                    base (option) {
                }
```

/* Dbset will be made for every Model. */

③ → public

```
Dbset <Model Name> Variable {get; set;}
```

/* Dbset should always be made public

Always make a primary key in the Model class before running migration, otherwise it won't create a migration.

④ Connection string:

→ connection string will be made in appsettings.json

"ConnectionStrings": {
"constr": "Server=.; database=DB; ~~trusted_connection~~ Trusted_Connection=true; TrustServerCertificate=true;"
}

⑤ Configure Program.cs

/* before var = builder.Build();

builder.Services.AddDbContext<~~DbContextClassName~~ (DbContextClassname)>
(item => item.UseSqlServer(builder.Configuration.GetConnectionString("constr")));

(what ever name was assigned while making the conn. str.)

⑥ Adding Migration. (to ~~add~~ make DB via C#)

In PM console => add-migration Name

Update-Database.

/* If you want to do any changes in the DB (column name or data type, etc), ~~you~~ do the changes in the Model class & run the migration again.

CRUD in DB:

To Display Data:

→ Making a controller. Just make a controller class in the controller folder. (You can use Home Controller aswell).

→ Whatever approach you might take, making a controller or using the Home controller you will need to set context.

→ Setting context means adding an object of the DbContext class in the controller because it contains DbSet which represents the table of the database.

public readonly Name of the Dbcontext class object;

/* Make a constructor.
public controllerclass( ){
this.object = object; }.

→ Then make an action method.
If you want to display data in the form of list, make a list in the action method.

<a class="btn btn-dark text-white" asp-action— to create a button.

25:00 (397)

public IActionResult Index() {
var students = object.DbSetName.ToList();
return view(students); }

Then generate Razor View of this by Right clicking and generating non-empty Razor View.cshtml

To create Data:

→ create an action method to create in the controller

public IActionResult Create(object std) {
if (ModelState.IsValid) {
context.Students.Add(std);
Object.

Context.SaveChanges();
return RedirectToAction("ViewName");
}
return View(std); }

Create the Details, Edit, & Delete methods similarly.