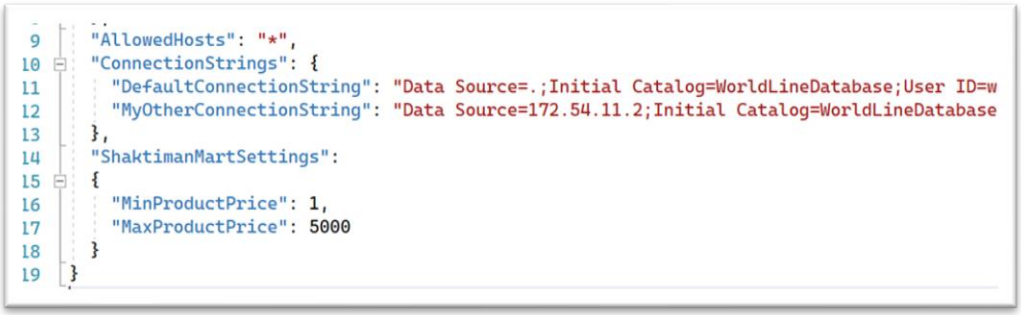
Revision

1. ViewData[“Title”]
    a. Create ViewData["Title"] in the view
    b. View is processed by asp.net
    c. View is merged with Layout.
    d. At this time since layout has <title>@ViewData["title"] - xyz </title>
    e. The title contents is replaced by the actual title specified in the View
2. Types of Persistent Stored in Asp.Net Core
    a. Application
        i. Is stored in server memory
        ii. Data is scoped to entire application, i.e accessible to all users
        Application["Count"]++;
        iii. This Count key is accessible from any user
    b. Session
        i. Is stored in server memory
        ii. Data is scoped to each user/session i.e. for one browser session with the server (multiple http transactions)
        Session["Count"] ++;
    c. ViewData
        i. Is stored inside each page in hidden input variables
        ii. Data is scoped only to current page instance. As long as you postback to same page or controller/action, the ViewData will be alive. If you navigate to different page, ViewData data is destroyed.
    d. TempData
        i. Is stored same as session
        ii. Data is scoped only from current Http Request till subsequent HttpRequest / till data is read
3. Steps to use IOptions service in Asp.NET Core
    a. Add the desired ConnectionStrings and other settings in appsettings.json



b. For each top level entry in appsettings.json, create a class

```
public class ConnectionStrings
{
    public string DefaultConnectionString { get; set; }
    public string MyOtherConnectionString { get; set; }
}

public class ShaktimanMartSettings
{
    public int MinProductPrice { get; set; }
    public int MaxProductPrice { get; set; }
}
```

c. In StartUp.cs, ConfigureServices method do the following

```
services.Configure<ConnectionStrings>(
      Configuration.GetSection("ConnectionStrings"));


services.Configure<ShaktimanMartSettings>(
       Configuration.GetSection("ShaktimanMartSettings"));
```

a. Now in any Service class or Controller class, let DI container send you the required settings as below

```
public HomeController(
            ILogger<HomeController> logger,
            IOptions<ConnectionStrings> connectionStringsAccessor,
            IOptions<ShaktimanMartSettings> shaktimanMartSettingsAccessor
            )
{
       …
}
```

a.

4. Entity Framework
   a. Install Nuget packages
      a. Microsoft.EntityFrameworkCore (ver 5)
      b. Microsoft.EntityFrameworkCore.SqlServer (ver 5)
      c. Or only Oracle.EntityFrameworkCore for Oracle Database (ver 5)
   b. Create a folder called [Data] and inside this create a class called ApplicationDbContext with constructor as shown

```
public class ApplicationDbContext : DbContext
{
       public ApplicationDbContext(
              DbContextOptions<ApplicationDbContext> options) : base(options)
       {
       }

       public DbSet<Student> Students { get; set; }
}
```

   c. In Startup.cs

```
services.AddDbContext<ApplicationDbContext>(options =>
options.UseSqlServer(Configuration.GetConnectionString("DefaultConnectionString"
)));
```

Or

```
services.AddDbContext<ApplicationDbContext>(opt       ions =>
options.UseOracle(Configuration.GetConnectionString("DefaultConnectionString"
)));
```

d. Your EF Core is ready to use as follows
    a. Inject the `ApplicationDbContext` in any controller or service class
    b. See some examples below
    c.
```
var students = await
_applicationDbContext.Students.ToListAsync();
```
    d.
```
await _applicationDbContext.Students.AddAsync(student);
await _applicationDbContext.SaveChangesAsync();
```
    e.
```
var studentFromDb = await
_applicationDbContext.FindAsync<Student>(rollNo);
```
    f.
```
_applicationDbContext.Update(student);
await _applicationDbContext.SaveChangesAsync();
```
    g.
```
var studentFromDb = await
_applicationDbContext.FindAsync<Student>(rollNo);
_applicationDbContext.Students.Remove(studentFromDb);
await _applicationDbContext.SaveChangesAsync();
```