Here’s a **step-by-step guide** to implement your Task Management ASP.NET Web API project with **log4net logging** and Entity Framework integration. I’ll structure it clearly so you can follow easily.

**1. Create the WebAPI Project**

Open terminal/command prompt:

dotnet new webapi -n dotnetapp

cd dotnetapp

Update the **launchSettings.json** to run on port 8080:

"applicationUrl": "http://0.0.0.0:8080"

Restore packages:

dotnet restore

**2. Install Required Packages**

**Entity Framework Core for SQL Server**

dotnet new tool-manifest

dotnet tool install --local dotnet-ef --version 6.0.6

dotnet add package Microsoft.EntityFrameworkCore.SqlServer

dotnet add package Microsoft.EntityFrameworkCore.Design

**log4net for Logging**

dotnet add package log4net

**3. Define Task Model**

Models/Task.cs:

using System;

namespace dotnetapp.Models

{

public class Task

{

public int Id { get; set; } // auto-incremented

public string Title { get; set; }

public string Description { get; set; }

public DateTime DueDate { get; set; }

}

}

**4. Setup DbContext**

Data/TaskDbContext.cs:

using Microsoft.EntityFrameworkCore;

using dotnetapp.Models;

namespace dotnetapp.Data

{

public class TaskDbContext : DbContext

{

public TaskDbContext(DbContextOptions<TaskDbContext> options) : base(options) { }

public DbSet<Task> Tasks { get; set; }

}

}

Add connection string in appsettings.json:

{

"ConnectionStrings": {

"DefaultConnection": "User ID=sa;password=examlyMssql@123;server=localhost;Database=appdb;trusted\_connection=false;Persist Security Info=False;Encrypt=False"

}

}

Configure Program.cs:

using dotnetapp.Data;

using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddDbContext<TaskDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection"))

);

var app = builder.Build();

app.UseAuthorization();

app.MapControllers();

app.Run();

**5. Configure log4net**

1. Add log4net.config in project root:

<?xml version="1.0" encoding="utf-8" ?>

<log4net>

<root>

<level value="DEBUG" />

<appender-ref ref="ConsoleAppender" />

</root>

<appender name="ConsoleAppender" type="log4net.Appender.ConsoleAppender">

<layout type="log4net.Layout.PatternLayout">

<conversionPattern value="%date %-5level %logger - %message%newline" />

</layout>

</appender>

</log4net>

1. In Program.cs add:

using log4net;

using System.Reflection;

log4net.Config.XmlConfigurator.Configure(new FileInfo("log4net.config"));

var logger = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

logger.Info("Application Started");

**6. Create TaskController**

Controllers/TaskController.cs:

using Microsoft.AspNetCore.Mvc;

using dotnetapp.Data;

using dotnetapp.Models;

using log4net;

namespace dotnetapp.Controllers

{

[ApiController]

[Route("[controller]")]

public class TaskController : ControllerBase

{

private readonly TaskDbContext \_context;

private static readonly ILog log = LogManager.GetLogger(typeof(TaskController));

public TaskController(TaskDbContext context)

{

\_context = context;

}

[HttpGet]

public IActionResult Index()

{

try

{

var tasks = \_context.Tasks.ToList();

log.Info("Tasks retrieved successfully.");

return Ok(tasks);

}

catch(Exception ex)

{

log.Error("Error retrieving tasks", ex);

return StatusCode(500, "Internal server error");

}

}

[HttpPost("create")]

public IActionResult Create(Task task)

{

try

{

\_context.Tasks.Add(task);

\_context.SaveChanges();

log.Info($"Task created successfully: {task.Title}");

return Ok(task);

}

catch(Exception ex)

{

log.Error("Error creating task", ex);

return StatusCode(500, "Internal server error");

}

}

[HttpPut("edit/{id}")]

public IActionResult Edit(int id, Task task)

{

try

{

var existingTask = \_context.Tasks.Find(id);

if(existingTask == null) return NotFound();

existingTask.Title = task.Title;

existingTask.Description = task.Description;

existingTask.DueDate = task.DueDate;

\_context.SaveChanges();

log.Info($"Task edited successfully: {task.Title}");

return Ok(existingTask);

}

catch(Exception ex)

{

log.Error("Error editing task", ex);

return StatusCode(500, "Internal server error");

}

}

[HttpDelete("delete/{id}")]

public IActionResult Delete(int id)

{

try

{

var task = \_context.Tasks.Find(id);

if(task == null) return NotFound();

\_context.Tasks.Remove(task);

\_context.SaveChanges();

log.Info($"Task deleted successfully: {task.Title}");

return Ok();

}

catch(Exception ex)

{

log.Error("Error deleting task", ex);

return StatusCode(500, "Internal server error");

}

}

}

}

**7. Create Initial Migration and Update Database**

dotnet dotnet-ef migrations add InitialSetup

dotnet dotnet-ef database update

**8. Run the Application**

dotnet run

Access endpoints:

* GET http://localhost:8080/task → List tasks
* POST http://localhost:8080/task/create → Create task
* PUT http://localhost:8080/task/edit/{id} → Edit task
* DELETE http://localhost:8080/task/delete/{id} → Delete task

All actions will be logged via **log4net**.