A **Generic Repository** in a Web API application is a design pattern that helps you write **reusable, clean, and maintainable data access logic** using Entity Framework (or any ORM). Instead of writing separate repository classes for every entity (like EmployeeRepository, ContactRepository, etc.), you can write one **generic** class that works with all entities.

**✅ Benefits**

* Reduces boilerplate code
* Promotes code reuse
* Centralizes data access logic
* Easy to test and maintain

**Structure of Generic Repository in a Web API**

**1. IGenericRepository Interface**

public interface IGenericRepository<T> where T : class

{

Task<IEnumerable<T>> GetAllAsync();

Task<T> GetByIdAsync(object id);

Task AddAsync(T entity);

void Update(T entity);

void Delete(T entity);

Task SaveAsync();

}

**2. GenericRepository Class**

public class GenericRepository<T> : IGenericRepository<T> where T : class

{

private readonly DbContext \_context;

private readonly DbSet<T> \_dbSet;

public GenericRepository(DbContext context)

{

\_context = context;

\_dbSet = context.Set<T>();

}

public async Task<IEnumerable<T>> GetAllAsync()

{

return await \_dbSet.ToListAsync();

}

public async Task<T> GetByIdAsync(object id)

{

return await \_dbSet.FindAsync(id);

}

public async Task AddAsync(T entity)

{

await \_dbSet.AddAsync(entity);

}

public void Update(T entity)

{

\_dbSet.Attach(entity);

\_context.Entry(entity).State = EntityState.Modified;

}

public void Delete(T entity)

{

\_dbSet.Remove(entity);

}

public async Task SaveAsync()

{

await \_context.SaveChangesAsync();

}

}

**3. Using It in a Controller**

[Route("api/[controller]")]

[ApiController]

public class EmployeeController : ControllerBase

{

private readonly IGenericRepository<Employee> \_employeeRepo;

public EmployeeController(IGenericRepository<Employee> employeeRepo)

{

\_employeeRepo = employeeRepo;

}

[HttpGet]

public async Task<IActionResult> GetAll()

{

var employees = await \_employeeRepo.GetAllAsync();

return Ok(employees);

}

[HttpPost]

public async Task<IActionResult> Add(Employee emp)

{

await \_employeeRepo.AddAsync(emp);

await \_employeeRepo.SaveAsync();

return Ok(emp);

}

}

**4. Registering in Program.cs / Startup.cs**

builder.Services.AddScoped(typeof(IGenericRepository<>), typeof(GenericRepository<>));