MONGO DB

İlişkisel olmayan veri tabanıdır.  
İlişkisel ile farkı çok daha hızlıdır , güncellik önemli değil ama hızlı bir veri akışı sağlanır genellikle sosyal medya gibi platformlarda da kullanılır.

|  |  |
| --- | --- |
| **İlişkisel** | **İlişkisel Olmayan** |
| **Tablolar** | **Collection** |
| **Satırlar** | **Document** |
| **Sutünlar** | **Field** |

**Veriler Bson veri tipiyle kaydedilir. Json formattadır ancak veri kaydedilirken Bson yani Binary JSON , binary şeklinde kayıt edilir hızını da burdan alır.**

**Kurulması gereken paketler :  
1)MongoDB.Bson  
2)MongoDB.Driver.Core  
3)MongoDB.Driver  
  
İdler ilişkiseldeki gibi int değil , string tutulur.  
ÖRNEK   
public class Category**

**{**

**[BsonId]**

**[BsonRepresentation(MongoDB.Bson.BsonType.ObjectId)]**

**public string categoryId { get; set; }**

**public string categoryName { get; set; }**

**}  
Not [BsonIgnore]  
BsonIgnore, MongoDB'nin C# driver'ında kullanılan bir attribute'dur. Bu attribute, bir sınıfın özelliklerinin MongoDB'ye yazılmasını engellemek için kullanılır. Yani, bu attribute ile işaretlenmiş bir özellik, MongoDB'ye veri yazılırken dikkate alınmaz ve veritabanına kaydedilmez.**

**Örneğin, MongoDB veritabanına veri gönderirken bazı özelliklerin kaydedilmesini istemeyebilirsiniz. Bu durumda, o özelliği BsonIgnore ile işaretlersiniz.**

**public class Product**

**{**

**[BsonId]**

**[BsonRepresentation(MongoDB.Bson.BsonType.ObjectId)]**

**public string productId { get; set; }**

**public string productCategoryId { get; set; }**

**[BsonIgnore]**

**public Category category { get; set; }**

**}**

**}**

**App Settings Ayarları:  
"DatabaseSettings": {**

**"CategoryCollectionName": "Categories",**

**"ProductCollectionName": "Products",**

**"ProductDetailCollectionName": "ProductDetails",**

**"ProductImageCollectionName": "ProductImages",**

**"ConnectionString": "mongodb://localhost:27017",**

**"DatabaseName": "MultiShopCatalogDb"**

**},**

**Bir klasör içinde bu ayarları interface ve sınıf yapılır.  
---**

**Entitiylerden nesne oluşturmak yerine entitiylerimizin propertilerimizi dtodaki propertilerle eşleştirmeye mapping denir.**

**ÖRNEK**

**public class GeneralMapping:Profile**

**{**

**public GeneralMapping() {**

**//Mapping for Category Entity**

**CreateMap<Category,ResultCategoryDto>().ReverseMap();**

**CreateMap<Category, CreateCategoryDto>().ReverseMap();**

**CreateMap<Category, UpdateCategoryDto>().ReverseMap();**

**CreateMap<Category, GetbyIdCategoryDto>().ReverseMap();**

**}**

**}**

**}**

**Bağlantı  
|  
Database  
|  
Tablo**

**using AutoMapper;**

**using MongoDB.Driver;**

**using MultiShop.Catalog.Dtos.CategoryDtos;**

**using MultiShop.Catalog.Entities;**

**using MultiShop.Catalog.Settings;**

**namespace MultiShop.Catalog.Services.CategoryService**

**{**

**public class CategoryService : ICategoryService**

**{**

**private readonly IMongoCollection<Category> \_categoryCollection;**

**private readonly IMapper \_mapper;**

**public CategoryService(IMapper mapper,IDatabaseSettings \_databaseSettings)**

**{**

**var client= new MongoClient(\_databaseSettings.ConnectionString);**

**var database = client.GetDatabase(\_databaseSettings.DatabaseName);**

**\_categoryCollection=database.GetCollection<Category>(\_databaseSettings.CategoryCollectionName);**

**\_mapper = mapper;**

**}**

**public async Task CreateCategoryAsync(CreateCategoryDto createCategoryDto)**

**{**

**var value=\_mapper.Map<Category>(createCategoryDto);**

**await \_categoryCollection.InsertOneAsync(value);**

**}**

**public async Task DeleteCategoryAsync(string id)**

**{**

**await \_categoryCollection.DeleteOneAsync(x => x.categoryId == id);**

**}**

**public async Task<List<ResultCategoryDto>> GetAllAsync()**

**{**

**var values=await \_categoryCollection.Find(x=>true).ToListAsync();**

**return \_mapper.Map<List<ResultCategoryDto>>(values);**

**}**

**public async Task<GetbyIdCategoryDto> GetbyIdCategoryAsync(string Id)**

**{**

**var value= await \_categoryCollection.Find<Category>(x=>x.categoryId==Id).FirstOrDefaultAsync();**

**return \_mapper.Map<GetbyIdCategoryDto>(value);**

**}**

**public async Task UpdateCategoryAsync(UpdateCategoryDto updateCategoryDto)**

**{**

**var values= \_mapper.Map<Category>(updateCategoryDto);**

**await \_categoryCollection.FindOneAndReplaceAsync(x=>x.categoryId==updateCategoryDto.categoryId,values);**

**}**

**}**

**}**