

# 第40期-集成MongoDB数据访问

2020年11月16日 9:19

## MongoDB 介绍

MongoDB 是一个基于分布式文件存储的 NoSQL 数据库，他支持的数据结构非常松散，存储类似格式类似于 JSON 文档，因此可存储比较复杂的数据类型，可扩展性也好，支持查询语言，可建立索引提高查询效率，缺点：对事务支持不够好，无法进行复杂的表级联，当然这是关系型数据库所擅长的，NoSQL 不是 SQL 的替代品，是弥补。

<https://baike.baidu.com/item/NoSQL>

<https://baike.baidu.com/item/mongodb>

<https://mongodb.net.cn>

<https://github.com/mongodb/mongo-csharp-driver>

[第30期 微服务使用NoSQL 数据库](#)

## ABP 与 MongoDB 集成

Install-Package Volo.Abp.MongoDB

[DependsOn(typeof(AbpMongoDbContext))]

## 创建 MongoDBContext 上下文

```
[ConnectionStringName("DogStore")]
public class DogMongoDbContext : AbpMongoDbContext
{
    public IMongoCollection<Dog> Dogs => Collection<Dog>();

    protected override void CreateModel(IMongoModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Dog>(b =>
        {
            b.CollectionName = "Dogs";
        });
    }
}

context.Services.AddMongoDbContext<DogMongoDbContext>(options =>
{
    options.AddDefaultRepositories(includeAllEntities: true);
});
```

```
});
```

## 映射扩展属性

```
public static class DogBsonClassMap
{
    private static readonly OneTimeRunner OneTimeRunner = new OneTimeRunner();

    public static void Configure()
    {
        OneTimeRunner.Run(() =>
        {
            BsonClassMap.RegisterClassMap<Dog>(map =>
            {
                map.ConfigureAbpConventions();
            });
        });
    }
}

public override void ConfigureServices(ServiceConfigurationContext context)
{
    DogBsonClassMap.Configure();
}
```

## 更改连接字符串

mongodb://[username:password@]host1[:port1][,host2[:port2],...[,hostN[:portN]]]/[database][?options]]

例如: mongodb://fred:foobar@localhost

例如: mongodb://localhost:27017

**现在可以在服务中注入并使用默认通用仓储了。**

## 替换默认仓储实现

```
options.AddRepository<Dog, DogMongoDbRepository>();
```

**替换后的好处是可以在自定义仓储中重写某方法。**

## 访问 MongoDB API 接口

```
IMongoDatabase database = _dogRepository.GetDatabase();
IMongoCollection<Dog> dogs = _dogRepository.GetCollection();
```

这种情况最好少用，破坏了抽象和封装。

## 事务支持

MongoDB 在 4.0 版本开始支持事务。

```
Configure<AbpUnitOfWorkDefaultOptions>(options =>
{
    options.TransactionBehavior = UnitOfWorkTransactionBehavior.Enabled;
});
```

## 实现并替换默认仓储基类

```
public class MyMongoDBRepositoryBase<TEntity> :
MongoDbRepository<DogMongoDbContext, TEntity> where TEntity : class, IEntity
{
    public MyMongoDBRepositoryBase(IMongoDbContextProvider<DogMongoDbContext>
dbContextProvider) : base(dbContextProvider)
    {
    }
}
```

```
public class MyMongoDBRepositoryBase<TEntity, TKey> :
MongoDbRepository<DogMongoDbContext, TEntity, TKey> where TEntity : class,
IEntity<TKey>
{
    public MyMongoDBRepositoryBase(IMongoDbContextProvider<DogMongoDbContext>
dbContextProvider) : base(dbContextProvider)
    {
    }
}
```

```
context.Services.AddMongoDbContext<DogMongoDbContext>(options =>
{
    options.SetDefaultRepositoryClasses(
        typeof(MyMongoDBRepositoryBase<,>),
        typeof(MyMongoDBRepositoryBase<>)
    );
});
```

## 抽象并提取上下文接口

```
public interface IDogMongoDbContext: IAbpMongoDbContext
{
    Collection<Dog> Dogs { get; }
```

```

}

context.Services.AddMongoDbContext<DogMongoDbContext>(options =>
{
    options.AddDefaultRepositories<IDogMongoDbContext>();
});

```

现在自定义仓储类也可以使用 `IBookStoreMongoDbContext` 接口了。

```

public class DogMongoDbRepository
    : MongoDBRepository<IDogMongoDbContext, Dog, int>,
        IDogRepository
{
    //...
}

```

## 替换现有的 DbContext

```

context.Services.AddMongoDbContext<OtherMongoDbContext>(options =>
{
    //...
    options.ReplaceDbContext<IDogMongoDbContext>();
});

```