

第15期-数据库初始化

2019年4月27日 00:53

数据库数据初始化

通过模型初始化数据

```
modelBuilder.Entity<Blog>().HasData(new Blog { BlogId = 1, Url =  
    "www.xcode.me" });
```

```
modelBuilder.Entity<Post>().HasData(new Post() { BlogId = 1, PostId = 1, Title =  
    "First post", Content = "Test 1" });
```

如果属性属于隐藏属性，无法点出属性，应该通过匿名对象初始化数据。

```
modelBuilder.Entity<Post>().HasData(new { BlogId = 1, PostId = 2, Title =  
    "Second post", Content = "Test 2" });
```

固有的实体类型可以以类似的方式进行初始化设定：

```
modelBuilder.Entity<Post>().OwnsOne(p => p.AuthorName).HasData(  
    new { PostId = 1, First = "Andriy", Last = "Svyryd" },  
    new { PostId = 2, First = "Diego", Last = "Vega" });
```

[完整的示例代码可参阅这里](#)

什么时候数据初始化才会被调用？ 1、迁移，2、context.Database.EnsureCreated()

手动迁移自定义项

HasData 转换为调用：InsertData(), UpdateData() 和 DeleteData(), 有时需要手动控制。

```
migrationBuilder.InsertData(  
    table: "Blogs",  
    columns: new[] { "Url" },  
    values: new object[] { "www.xcode.me" });
```

自定义初始化逻辑

```
using (var context = new DataSeedingContext())  
{  
    context.Database.EnsureCreated();
```

```
        var testBlog = context.Blogs.FirstOrDefault(b => b.Url ==  
"www.xcode.me");  
        if (testBlog == null)  
        {  
            context.Blogs.Add(new Blog { Url = "www.xcode.me" });  
        }  
        context.SaveChanges();  
    }
```