

第34期-使用MediatR实现CQRS模式

2021年6月17日 21:16

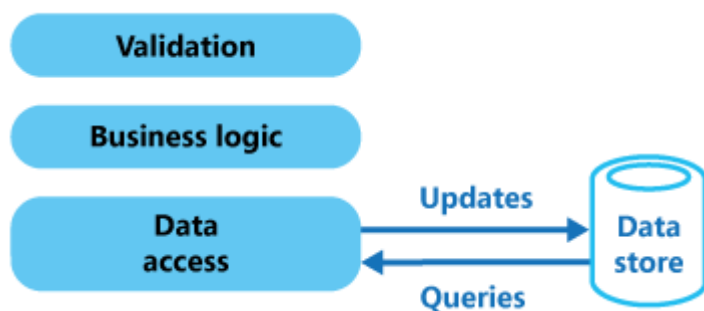
什么是 CQRS 模式？

CQRS 是“命令查询责任分离”（Command Query Responsibility Segregation）的缩写。

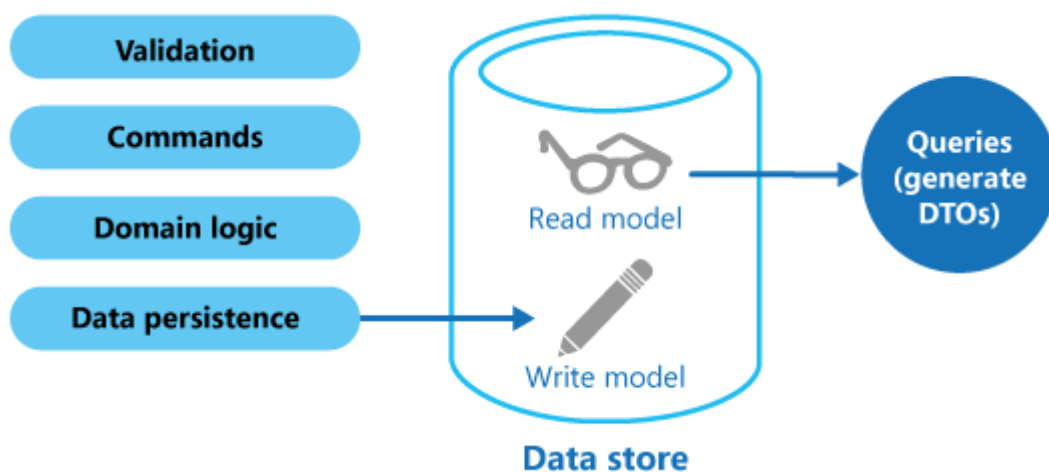
CQRS 命令和查询责任分离数据存储读取和更新操作的模式。在应用程序中实现 CQRS 可以最大程度地提高其性能、可伸缩性和安全性。通过迁移到 CQRS 而创建的灵活性使系统能够更好地随着时间的推移更好地发展，并防止更新命令在域级别导致合并冲突。

传统的混合模式

在传统的体系结构中，使用同一数据模型查询和更新数据库，读取和写入工作负荷通常是非对称的，其性能和缩放要求非常不同。



命令和查询分离



可缩放的读写分离



读写分离的优点

独立缩放，优化的数据架构，安全性，关注点分离，具体化查询更简单。

面临的问题和挑战

复杂性，消息同步可靠性，最终一致性时效性。

在零度框架中实现 CQRS 模式

使用 Dapper 灵活查询数据，使用 Mediator 实现命令模式。

Install-Package Dapper

Install-Package MediatR.Extensions.Microsoft.DependencyInjection

命令消息去重机制的设计与实现

```
public IActionResult Error()  
{  
    return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??  
    HttpContext.TraceIdentifier });  
}
```

[Should I use Request-Id, X-Request-Id or X-Correlation-Id in the request header?](#)

[How is HttpContext TraceIdentifier generated in .NET Core?](#)

[Building End-to-End Diagnostics and Tracing: Trace Context](#)

[Improvements in .NET Core 3.0 for troubleshooting and monitoring distributed apps](#)

Azure 云架构设计模式

<https://docs.microsoft.com/zh-cn/azure/architecture/patterns/cqrs>