Golang □□□□ *

- □□□ Limit □□ □□

개선된 Limit 사용 방식

1. 기존 방식 : Limit와 IsntPagination

기존의 백엔드 설계 방식에서는 모든 데이터를 가져오기 위해 다음과 같이 설계되었습니다

- `QueryVars.IsntPagination = true`를 설정하여 페이지네이션을 비활성화
- `vRet.PageVars.Limit` 값을 매 `1000000000`)로 설정

하지만 이러한 접근 방식은 다음과 같은 문제가 있었습니다

- 코드 가독성 저하
- 불:요] `Limit` 값은 메모리와 리소스의 낭비를 발생 시킬 수 있습니다.

2. 개선된 방식

`IsntPagination` ListType1 파라 `Limit = 0` 으로 설정하면 모든 데이터를 한 번에 가져오도록 설계되었습니다

(Limit > 500 이상이면 **0**으로 처리됩니다**)**

개선 내용 **:**

- 코드 `IsntPagination` ListType1 파라미터 옵션을 제거하여 설정을 간소화.
- 자원 `Limit = 0` 으로 동작을 제어하므로 메모리 사용이 최적화됨.
- 유지보수 편의성: 직관적인 코드 구조로 인해 개발자가 쉽게 이해하고 수정 가능.

3. Query File에서의 Limit 사용

페이징 `LIMIT` [`OFFSET`]을 명시적으로 설정하면 됩니다. (페이징 처리가 불필요할 경우 안넣어줘도 됩니다)

```
SELECT
    mx.id as id,
    turbo_thumb as c1,
    item_name as c2,
    item_slug as c3
FROM dbr_item as mx
    INNER JOIN dbr_igroup as mb ON mx.igroup_id = mb.id
-- @where
limit 1
```

`LIMIT` 은 반환[**`OFFSET`**)와 개수 를 계산하기 때문에 추가적인 연산 작업이 발생하기 때문에 쿼리 속도가 늦어집니다.