□□□ - □□□ Quick Start

- □□□ □□ □□□
- □□□ main_api □□□□□
- MAIN_API□ □□□□□ □□□□
- Golang □□□□ *
    - □□□ Limit □□ □□
- DB□□ ListToken □□
- □□□ □□□ □□□ □□ □□ □□ (Kafka, ES)
- Echo□□ HTTP □□ □□

□□□ □□ □□□

□□□ □□□□□□ □□□□□□ □□□ □□□□

Router -> Middleware -> Controller -> Handler(Controller□ Model□ □□ □□□□□□ □□□□□ □□ □□□□□ □□□□ □□) -> Mo
□□□.

□□□ main_api □□□□□

MAIN_API □□ □□!

1. WSL□ □□□□□.

2. □□□□ □□□□□ □□□□ □□□ □□□□□.

   □) sudo adduser <username>.

3. home □□□ □□□ □□□□ □□□ □□□□□ □□□□□.

   cd /home

   mkdir main_api

4. □□□ □□□ main_api(cd main_api) □. (

5. wget http://dbr02-wget.daboryhost.com/main_api.tar.gz □ □□

   (wget□ □□□□□□ □□□ □□□□□□ □□ □□)

□□□ □□□□□□ □□ □□□□□□ □□□□□ □□□□□□.

(1). cache-key-pair

(2). weberp-go

(3). weberp-queries

(4). mybin

6. 파일들이 전부 존재하면 디렉토리 접근 후 실행파일 가동하여 서버 시작합니다

```
1. cd weberp-go
```

```
2. ./weberp-go
```

MAIN_API[illegible] [illegible] [illegible]

Docker([illegible])[illegible] [illegible]([illegible])[illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

.env.dabory
MAIN_API_URL='http://host.docker.internal:18080' <- URL[illegible] [illegible] [illegible].

MAIN_API_URL = host.docker.internal:([illegible])

[illegible] [illegible] Docker [illegible] [illegible], [illegible] [illegible] [illegible]([illegible] [illegible])[illegible] [illegible] [illegible]

- Docker [illegible] `localhost`[illegible] [illegible].
- [illegible] Docker [illegible] [illegible], [illegible] [illegible] [illegible]([illegible] [illegible])[illegible] [illegible] [illegible].

MAIN_API_URL= localhost:([illegible])

Docker[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

ThunderClient[illegible] [illegible] [illegible]

url = http://localhost:18080/gate-token-get
body[illegible] [illegible] [illegible] [illegible] (ClientId, Base64) [illegible] [illegible] [illegible] -> [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] -> [illegible] [illegible] [illegible]

Golang □□□ *

[illegible] Limit [illegible]

1. [illegible] : Limit[illegible] IsntPagination

[illegible]

- `QueryVars.IsntPagination = true` [illegible]
- `vRet.PageVars.Limit` [illegible] `1000000000`)[illegible]

[illegible]

- [illegible]
- [illegible] `Limit` [illegible].

2. [illegible]

`IsntPagination` ListType1 [illegible] `Limit = 0` [illegible]. [illegible]

(Limit > 500 [illegible] 0[illegible])

[illegible] :

- [illegible] `IsntPagination` ListType1 [illegible].
- [illegible] `Limit = 0` [illegible].
- [illegible].

3. Query File[illegible] Limit [illegible]

[illegible] `LIMIT` [illegible] `OFFSET` [illegible]. ([illegible])

```
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` □ □□□ (**`OFFSET`**) □□□□□□□□ □□□□ □□□□□ □□ □□□□ □□□□□ □□□□ □□ □□□□ □□□□□□.

DB□□ ListToken □□

ListToken □□□?

- DbtListType1 □**ListToken** □□□□□□□ □□
- □□ □□□ DbtListType1 □□□□□□□□□□□□□□□□
- □□□**ListToken**□□□□ □□□□□□□□

ListToken□ □□ □□

1. □□□□DbtListType1 □□□□□□□□□□□□□□
2. □□□□**ListToken**□□□ □□□□
3. □□□□□□□□ □□□□ □□ □□□□□ □□□□ □□**(Pick, Act, Page □)**□ □□ □□

□□ □□

□□	□□
□□ □□ □□□	□□ □DbtListType1□□□□
ListToken □□	□ListToken□ □□ □□ □□□□□ □□ □□□□ □□
□□□ □□□	□ □Pick□, □Act□, Page□ □□ □□□□ □□

□ □□□□□ □□ □□□□ □□□□ □□□□□ □□□□, □□□□□ □□□□ □□□□□□ □□□□ □ □□□□ □□□□

[illegible] (Kafka, ES

[illegible] Kafka + Bulk [illegible]

[illegible]

1. **DB -> Kafka -> ElasticSearch** [illegible]
2. **Kafka**[illegible] **bucket** [illegible]
3. **ElasticSearch**[illegible] **Bulk API**[illegible]

[illegible]

1. **Kafka**[illegible] **Bucket** [illegible] → [illegible]
2. **ElasticSearch**[illegible] **Bulk** [illegible] → [illegible]
3. [illegible] **(150,000** [illegible] **)** [illegible]

Kafka Bucket [illegible]

```
qryCnt := models.QryCount(y.Db, cntStr)
	fmt.Println("qryCnt:", qryCnt)


	bucketCnt := 0
	if qryCnt == 0 {
		bucketCnt = 0
	} else {
		bucketCnt = 1 + (int(qryCnt) / p.Limit)
	}


    fmt.Println("bucketCnt:", bucketCnt)
	for i := 0; i < bucketCnt; i++ {
		// for i := 0; i < 3; i++ {
		p.Offset = p.Limit * i
		sqlStr, _, _ := models_join.ListType1PlainQryStrGet(y, lt, p, q, "/list")
		if err := t.ProduceaBucketToKafka(y, "kkk", headers, sqlStr, topic); err != nil {
```

```
		return errors.New(e.PageQryErr("vrt452", ": "+err.Error()))
	}
}
```

Bucket을 활용하여 병렬 처리 가능**!**

ElasticSearch Bulk API 예시

```
for _, row := range v.Page {
	row.ClientCode = clientCode
	fmt.Printf("BuyerId: %d, LastSorderDate: %s, ClientCode : %s\n", row.Id, row.LastSorderDate, row.ClientCode)
	row.LastVistDate = row.LastSorderDate
	// Meta 데이터 작성
	meta := fmt.Sprintf(`{ "index": { "_index": "%s", "_id": "%d" } }%s`, esIndex, row.Id, "\n")
	bulkRequest.WriteString(meta)

	// 문서 데이터 작성
	data, _ := json.Marshal(row)
	bulkRequest.WriteString(string(data) + "\n")
	fmt.Println(string(data)) // 데이터를 출력하여 검증

	fmt.Printf("Prepared document for BuyerId: %d\n", row.Id)
}

// Bulk 요청 전송
res, err := es.Bulk(bytes.NewReader(bulkRequest.Bytes()))
if err != nil {
	return fmt.Errorf("bulk request failed: %w", err)
}
defer res.Body.Close()

// Elasticsearch 응답 확인
if res.IsError() {
	log.Printf("Bulk indexing failed: %s", res.String())
	return fmt.Errorf("bulk indexing failed")
}
```

Bulk API를 활용하면 한 번에 많은 데이터를 삽입 가능**!**

[illegible]

Echo□□ HTTP □□ □□

1. □□

Dabory□**Go** □□□□□ **Echo** □□□□□□ □□□□□ □□ □□□□□□ **Json** [`c.String()` [`c.JSONBlob()` □ □□□□ HTTP □□□ Json□□ □□□□ [

---

2. `c.String(statusCode int, response string)` □□□

□ □□

- `c.String()` □□□□□□□□□□□□□□. □□
- `statusCode` □ HTTP □ `response` □ □□□□□□□ □□ □□ □□□□.

□ □□ □□

var c echo.Context
`return c.String(607, "")`

□ □□ □□

- `607` □□□□□ **HTTP**□ □□□□□□□□□□□.
- □ `""` □□ □□□□)□□.
- [**`response.status`** [**`607`** □□. □□ □□

---

3. `c.JSONBlob(statusCode int, jsonData []byte)` □□□

□ □□

- `c.JSONBlob()` □**JSON** □□□□□□□□□□□□
- `statusCode` □ HTTP □ `jsonData` □ □□ □□□□ J `[]byte`)□□□□.

□ □□ □□

```
ret, err := json.Marshal(vRet)
if err != nil {
    return c.String(500, "Failed to encode JSON: "+err.Error())
}
return c.JSONBlob(http.StatusOK, ret)
```

□ □□ □□

1. `json.Marshal(vRet)`을 `ZbaksanSorderEyetestRes` 구조체를 JSON 데이터로 변환한다.
2. 변환된 JS`c.JSONBlob(http.StatusOK, ret)`을 통해 프론트엔드로 전송한다.
3. [**`response.json()`** [**`response.data`**로 **JSON**을 받을 수 있다.

---

4. `c.String()` vs `c.JSONBlob()` 비교

기능	**`c.String()`**	**`c.JSONBlob()`**
응답 형식	문자열**(String)**	**JSON (**바이트 배열**)**
사용 목적	단순 메시지, 상태 코드 반환	JSON 데이터 응답
예제 코드	`c.String(607, "")`	`c.JSONBlob(http.StatusOK, ret)`
프론트에서 처리	`response.text()`	`response.json()` [`response.data`

---

5. 요약

[**`c.String(607, "")`**

- HTTP 상`607`을 반환하여 프론트엔드에서 감지 가능.
- [`response.status` 값으로 확인 후 특정 로직 실행.

[**`c.JSONBlob(http.StatusOK, ret)`**

- JSON 데이터를 반환할 때 사용.
- `json.Marshal()`을 통해 JSO`response.json()`으로 처리.