

# Echo HTTP 方法

## 1. 概要

Dabory Go 標準ライブラリ **Echo** は HTTP 応答を生成するための **Json** 形式の `c.String()` `c.JSONBlob()` などの HTTP 応答を生成するための方法を提供しています。

---

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

概要

- `c.String()` は `statusCode` と `response` を生成します。
- `statusCode` は HTTP ステータスコード、`response` は応答の文字列です。

例

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

コード

- `607` は HTTP ステータスコードです。
  - `""` は応答の文字列です。
  - `response.status` は `607` のステータスコードです。
- 

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

概要

- `c.JSONBlob()` は **JSON** 形式の応答を生成します。
- `statusCode` は HTTP ステータスコード、`jsonData` は応答の `[]byte` です。



□ □ □

```
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)` □ `ZbaksanSorderEyetestRes` □ □ □ □ JSON □ □ □ □.
2. □ □ □ `c.JSONBlob(http.StatusOK, ret)` □ □ □ □ □ □ □ □.
3. □ `response.json()` □ `response.data` □ **JSON** □ □ □ □.

## 4. `c.String()` vs `c.JSONBlob()` □ □

□ □	<code>c.String()</code>	<code>c.JSONBlob()</code>
□ □ □	<code>text(String)</code>	<b>JSON</b> (□ □ □)
□ □ □	□ □ □ □, □ □ □ □	JSON □ □ □
□ □ □	<code>c.String(607, "")</code>	<code>c.JSONBlob(http.StatusOK, ret)</code>
□ □ □ □	<code>response.text()</code>	<code>response.json()</code> □ <code>response.data</code>

## 5. □ □

□ `c.String(607, "")`

- HTTP □ `607` □ □ □ □ □ □ □ □ □ □.
- □ `response.status` □ □ □ □ □ □ □ □.

□ `c.JSONBlob(http.StatusOK, ret)`

- JSON □ □ □ □ □ □.
- `json.Marshal()` □ □ □ JSON `response.json()` □ □ □.



