
















Echo HTTP

1.

--	--

DaboryGo 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`  `[]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)` □ `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>	JSON (□ □ □ □)
□ □ □	□ □ □ □, □ □ □ □	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()` □ □ □ □.

