[WebApp] Protocal22-API 사용예[(GoLang)

> [WebApp] Protocal22-API 사용예제2

시작하기 전에

이 문서는 Dabory OwnerKey API for WebApp을 사용해 main app과 guest app 개발시 필요한 참고 정보를 안내합니다.

WebApp 사용예제

2. WebApp 사용 예제

□□ GateToken 발급 스케줄러

```
package main
func main() {
	// 새로운 고루틴(Goroutine) 실행: 백그라운드에서 주기적으로 실행됨
	go func() {
		for {
			// 1. UpdateGateTokenIfNeeded() 함수 호출 (GateToken 발급 및 업데이트)
			err := controllers_func.UpdateGateTokenIfNeeded()
```

```
			
			// 2. 만약 에러가 발생하면 로그를 남김
			if err != nil {
				e.ErrLog("UpdateGateTokenIfNeeded Error", err)
			}

			// 3. 2초 동안 대기 후 반복 (즉, 2초마다 실행됨)
			time.Sleep(2 * time.Second)
		}
	}()

	// 주의: main 함수가 종료되면 고루틴도 종료됨
	select {} // main 함수가 종료되지 않도록 무한 대기
}
```

"OwnerKey"를 활용한 GateToken, BackUrl 반환

Host의 dabory-app/gate-token-get을 호출 -> Host -> main_api (GateToken) 반환

```
// GateToken을 요청하는 함수
func UpdateGateTokenIfNeeded() error {
	myOwnerKey := "My_OwnerKey" //OwnerKey (GateToken 발급 주체 선정을 위해 사용)

	// GateToken을 요청 (OwnerKey를 사용하여 요청)
	err := locals.GuestGateTokenGets(myOwnerKey)
	if err != nil {
		fmt.Println("GateToken 요청 실패:", err)
		return err
	}

	return nil
}



type AppApi struct {
	ApiUrl    string
	GateToken string
}
```

```
var GAppApis [2]AppApi // GateToken, ApiUrl □□ □□

func GuestGateTokenGets(myOnwerKey string) error {
□appType := 0 //Dbupdate
□// 0:keypair, 1:Dbu  // "SsoConnString" Must be Deprecated

□var err error

□if myOnwerKey != "" {
□□// e.LogStr("lskdjqfals", e.LogFuncName()+"; GuestGateTokenOwner with OwnerKey: "+myOnwerKey)
□□_, _, err = GuestGateTokenOwner(appType, "https://host_domain_URL.com", myOnwerKey) // app type ->
OwnerCode
    // Host Domain URL, OwnerKey□ □□□□□ □□
□□if err != nil {
□□□return e.ErrLog(e.FuncRun("23rfsr3qrase", e.CurrFuncName()), err)
□□}
□}

□return nil
}

func GuestGateTokenOwner(appType int, pivotUrl string, ownerKey string) (string, string, error) { // HOST
URL/dabory-app ->  pivot URL
□var appTypeCode string
□if appType == 0 {
□□appTypeCode = "keypair"
□} else if appType == 1 {
□□appTypeCode = "dbupdate"
□}

□if GAppApis[appType].GateToken != "" { // □□□□ □□ □□ (□□ GateToken□ □□ □□)
□□// fmt.Println("34092ujfa : "+"Using GateToken in ARRAY to access : ", GAppApis[0].GateToken)
□} else { // (GateToken□ □□ □□ □□ □□)
□□fmt.Println("34092ujfa : " + "isn't exist GateToken in ARRAY to access ")

□□vGt := &GateTokenGetReq{
□□□OwnerKey: ownerKey,
□□}
□□bodyBytes, _ := json.Marshal(vGt)
```

□□frontUrl := pivotUrl + "/dabory-app/gate-token-get"
□□msgBytes, staInt, err := HttpResponseSimplePost("POST", frontUrl, bodyBytes)
// Strong -> Host Lalavel□ Http □□ -> Lalavel -> main_api□ GateToken □□
□□e.LogStr("23rfsr3qr-GateTokenGetReq", e.LogFuncName()+"; Raravel frontUrl : "+frontUrl)
□□if err != nil {
□□□return "", "", e.ErrLog(e.FuncRun("45425fd34sd-The HTTP request "+frontUrl, e.CurrFuncName()), err)
□□}

□□if staInt != 200 {
□□□TrackFailure() // □□ □□ □ □□ □□ □□
□□□fmt.Println("GateToken □□ □□:", err, "(□□ 1□□ □ □□ □□:", len(failTimestamps), ")", "□□ Code : ", staInt)

□□□// □□ 1□□ □□ □□ □□□ maxFailures(3) □□□□ □□□□ □□
□□□if len(failTimestamps) >= maxFailures {
□□□□fmt.Println("1□□ □ GateToken □□□", maxFailures, "□ □□ -> □□□□□ □□")
□□□□os.Exit(1) // □□□□ □□
□□□}

□□□return "", "", errors.New(e.FuncRun("87ty344ra3-Request Fail "+string(msgBytes), e.CurrFuncName()))
□□}

□□failTimestamps = []int64{} // □□ □ □□ □□ □□□

□□ret := &struct {
□□□ApiUrl   string
□□□GateToken string
□□}{}

□□if err := json.Unmarshal(msgBytes, ret); err != nil { // GateToken□ BackendUrl□ ret □□□□ □□
□□□return "", "", e.ErrLog(e.FuncRun("45425fd34sd-Json Format "+frontUrl, e.CurrFuncName()), err)
□□}
// □□ □□ GateToken□ □□ □□□ □□
□□GAppApis[appType].ApiUrl = ret.ApiUrl
□□GAppApis[appType].GateToken = ret.GateToken
□□e.OkLog("Just Added GateToken in ARRAY to access AppType: " + appTypeCode)
□□e.OkLog("Just Added GateToken in ARRAY to access AppType: " + ret.GateToken)
□□e.OkLog("Just Added GateToken in ARRAY to access AppType: " + ret.ApiUrl) // main_api URL

□}

```
□return GAppApis[appType].ApiUrl, GAppApis[appType].GateToken, nil
}
```

□□ □□□□ GateToken□ □□□□ main_api □□

```
// □□ API □□□ □□□□ □□
func process() {
□// 1. GateToken□ API URL □□□□
□apiUrl := locals.GAppApis[0].ApiUrl    // API □□□ □□ □□ URL
□gateToken := locals.GAppApis[0].GateToken // API □□ □ □□□ □□ □□

□// 2. □□□ API □□□□□ □□ (credit-page □□)
□mainApiURL := apiUrl + "/credit-page"

□// 3. □□ □□ □□ (□□□ □□□)
□requestBody := map[string]interface{}{
□□"PageVars": map[string]interface{}{
□□□"Fields":     "credit_no",
□□□"Limit":      1,
□□□"Desc":       "id",
□□□"MyFilter":   "",
□□□"QueryCnt":   0,
□□□"Query":      "",
□□□"Asc":        "",
□□□"Offset":     0,
□□□"ReturnJson": "",
□□},
□}

□// 4. API □□ □□ (GateToken □□)
□Body, err := RequestWithGateToken(mainApiURL, gateToken, requestBody)
□if err != nil {
□□// □□ □□ □ □□ □□ □□
□□e.ErrLog("API □□ □□: ", err)
□□return
□}
```

```
□// 5. □□ □□□ □□
□fmt.Println("API □□ □□□:", string(Body))
}
```

□□ □□□ GateToken□□ □□ (□□□ GateToken □ □□)

```
// GateToken□ □□□□□ API □□□ □□□ □□
func RequestWithGateToken(mainApiURL string, gateToken string, requestBody map[string]interface{}) ([]byte,
error) {
□// 1. HTTP □□ □□□ (□□ GateToken □□)
□resp, body, err := SendHttpRequest(mainApiURL, gateToken, requestBody)
□if err != nil {
□□fmt.Println("HTTP □□ □□:", err)
□□return nil, err
□}
□defer resp.Body.Close()

□fmt.Println("□□ □□:", string(body)) // API □□ □□

□// 2. □□ □□□ 505□□ -> GateToken □□ (□□□ GateToken □□ □□)
□if resp.StatusCode == 505 {
□□fmt.Println("GateToken □□! □□□ GateToken □□ □□..")

□□// □□ □□□ GateToken □□□
□□locals.GAppApis[0] = locals.AppApi{}

□□// □□□□□ □□□ GateToken□ □□□ □□□ □□ (□□ 5□□□ □□)
□□for locals.GAppApis[0].GateToken == "" {
□□□fmt.Println("□□□ GateToken □□ □..")
□□□time.Sleep(5 * time.Second)
□□}

□□fmt.Println("□□□ GateToken □□ □□!")

□□// □□□ GateToken□□ API □□□
□□newGateToken := locals.GAppApis[0].GateToken
□□resp, body, err := SendHttpRequest(mainApiURL, newGateToken, requestBody)
```

□□if err != nil {
□□□fmt.Println("HTTP □□ □□:", err)
□□□return nil, err
□□}
□□defer resp.Body.Close()

□□// □□□ □□ □□ □□
□□if resp.StatusCode == 200 {
□□□e.OkLog("□□□ GateToken□ □□□ HTTP □□ □□!")
□□□return body, nil
□□} else {
□□□return nil, fmt.Errorf("□□□ GateToken □□ □□, □□ □□: %d", resp.StatusCode)
□□}
□}

□// □□ GateToken□□ □□ □□□ □□
□e.OkLog("□□ GateToken□ □□□ HTTP □□ □□!")
□return body, nil
}

// HTTP □□□ □□□ □□
func SendHttpRequest(mainApiURL string, gateToken string, requestBody map[string]interface{})
(*http.Response, []byte, error) {
□// 1. JSON □□□□ □□
□jsonData, err := json.Marshal(requestBody)
□if err != nil {
□□fmt.Println("JSON □□ □□:", err)
□□return nil, nil, err
□}

□// 2. HTTP □□ □□ (POST □□)
□req, err := http.NewRequest("POST", mainApiURL, bytes.NewBuffer(jsonData))
□if err != nil {
□□fmt.Println("HTTP □□ □□ □□:", err)
□□return nil, nil, err
□}

□// 3. HTTP □□ □□ (GateToken □□)
□req.Header.Set("Content-Type", "application/json")
□req.Header.Set("GateToken", gateToken)

```
    // 4. HTTP □□ □□
    client := &http.Client{}
    resp, err := client.Do(req)
    if err != nil {
        fmt.Println("HTTP □□ □□:", err)
        return nil, nil, err
    }

    // 5. □□ □□□ □□
    body, err := io.ReadAll(resp.Body)
    if err != nil {
        fmt.Println("□□ □□ □□ □□:", err)
        return nil, nil, err
    }

    // 6. □□ □□ □□
    fmt.Println("□□ □□:", resp.StatusCode)
    fmt.Println("□□ □□:", string(body))

    return resp, body, nil
}
```

Revision #15
Created 3 March 2025 10:04:16 by □□□
Updated 20 March 2025 12:05:55 by □□□