

[WebApp] Protocol22-API 实现 (GoLang)

“

[WebApp] Protocol22-API 实现2

实现 实现

实现 Dabory OwnerKey API for WebApp 实现 main app 实现 guest app 实现 实现 实现 实现.

WebApp 实现

2. WebApp 实现 实现

实现 GateToken 实现 实现

```
package main

func main() {
    // 实现 实现(Goroutine) 实现: 实现 实现 实现
    go func() {
        for {
            // 1. UpdateGateTokenIfNeeded() 实现 (GateToken 实现 实现)
            err := controllers_func.UpdateGateTokenIfNeeded()
```

```

    }

    // 2. 1초 뒤 GateToken 갱신
    if err != nil {
        e.ErrLog("UpdateGateTokenIfNeeded Error", err)
    }

    // 3. 2초 뒤 GateToken 갱신 (1, 2초 뒤)
    time.Sleep(2 * time.Second)
}

}()

// 4. main 함수 실행
select {} // main 함수 실행
}

```

4. "OwnerKey"로 GateToken, BackUrl

Host는 dabory-app/gate-token-get으로 Host -> main_api (GateToken)로

```

// GateToken을 갱신
func UpdateGateTokenIfNeeded() error {
    myOwnerKey := "My_OwnerKey" //OwnerKey (GateToken을 갱신)

    // GateToken을 갱신 (OwnerKey를 사용)
    err := locals.GuestGateTokenGets(myOwnerKey)
    if err != nil {
        fmt.Println("GateToken 갱신:", 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("Iskdjqfals", 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("23rfsr3qrse", 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, stalnt, 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 stalnt != 200 {
        TrackFailure() // 
        fmt.Println("GateToken : ", err, "(1 : ", len(failTimestamps), ")", " Code : ", stalnt)

        // 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. API 호출
```

```
fmt.Println("API 호출:", string(Body))  
}
```

Gate Token을 받아서 (Gate Token을 받아서)

```
// Gate Token을 받아서 API 호출
```

```
func RequestWithGateToken(mainApiURL string, gateToken string, requestBody map[string]interface{}) ([]byte, error) {
```

```
// 1. HTTP 호출 (Gate Token을 받아서)
```

```
resp, body, err := SendHttpRequest(mainApiURL, gateToken, requestBody)
```

```
if err != nil {
```

```
    fmt.Println("HTTP 호출:", err)
```

```
    return nil, err
```

```
}
```

```
defer resp.Body.Close()
```

```
fmt.Println("결과:", string(body)) // API 호출 결과
```

```
// 2. 505 응답 -> Gate Token을 받아서 (Gate Token을 받아서)
```

```
if resp.StatusCode == 505 {
```

```
    fmt.Println("Gate Token을 받아서! Gate Token을 받아서.")
```

```
// Gate Token을 받아서
```

```
locals.GAppApis[0] = locals.AppApi{}
```

```
// Gate Token을 받아서 (5초 대기)
```

```
for locals.GAppApis[0].GateToken == "" {
```

```
    fmt.Println("Gate Token을 받아서.")
```

```
    time.Sleep(5 * time.Second)
```

```
}
```

```
fmt.Println("Gate Token을 받아서!")
```

```
// Gate Token을 받아서 API 호출
```

```
newGateToken := locals.GAppApis[0].GateToken
```

```
resp, body, err := SendHttpRequest(mainApiURL, newGateToken, requestBody)
```

```

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

    // 检查状态码
    if resp.StatusCode == 200 {
        e.OkLog("成功获取GateToken HTTP 200!")
        return body, nil
    } else {
        return nil, fmt.Errorf("成功获取GateToken 失败, 状态码: %d", resp.StatusCode)
    }
}

// 获取GateToken
e.OkLog("获取GateToken HTTP 200!")
return body, nil
}

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

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

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

```

```

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

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

// 6. 打印响应
fmt.Println("状态码:", resp.StatusCode)
fmt.Println("响应体:", string(body))

return resp, body, nil
}

```

Revision #15

Created 3 March 2025 10:04:16 by 用户

Updated 20 March 2025 12:05:55 by 用户