博客系统

功能：主页、注册、登录、发表文章、为文章点赞、删除文章、留言、删除留言

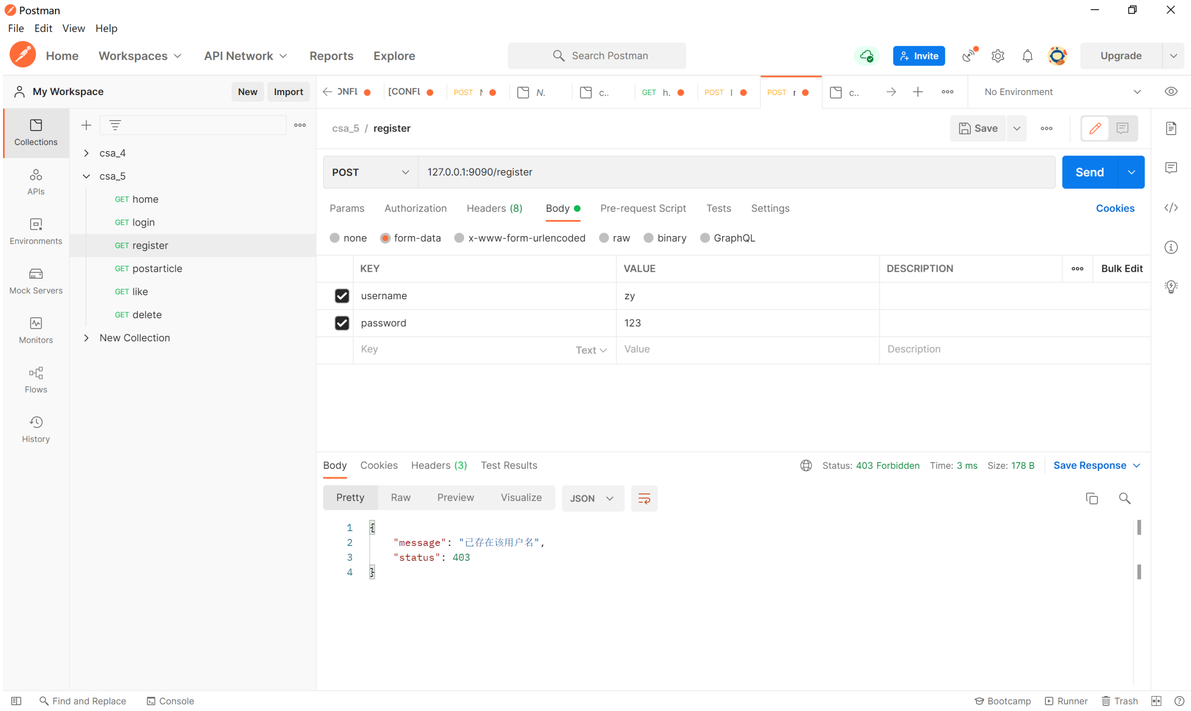
1.注册 127.0.0.1:9090/register post请求

```
r.POST("/register", controller.Register)
```

```
//Register
//@title       Register()
//@description 注册请求
//@author      zy
//@param       c *gin.Context
//@return
func Register(c *gin.Context) {
   var u userinformation.UserInfo
   err := c.ShouldBind(&u)             //参数绑定
   if err != nil {
      c.JSON(http.StatusOK, gin.H{
         "code": 2001,
         "message": "无效的参数",
      })
      return
   }

   //判断用户名是否已经存在
   if common.QueryUserInfo(u) {
      c.JSON(http.StatusForbidden, gin.H{
         "status": http.StatusForbidden,
         "message": "已存在该用户名",
      })
      return
   } else {
      if common.InsertUserInfo(u){
         c.JSON(http.StatusOK, gin.H{
            "status": http.StatusOK,
            "message": "注册成功",
         })
         return
      } else {
         c.JSON(http.StatusInternalServerError, gin.H{
            "status": http.StatusInternalServerError,
            "message": "some errors in sql",
         })
      }
   }
}
```

测试图


csa_5 / register

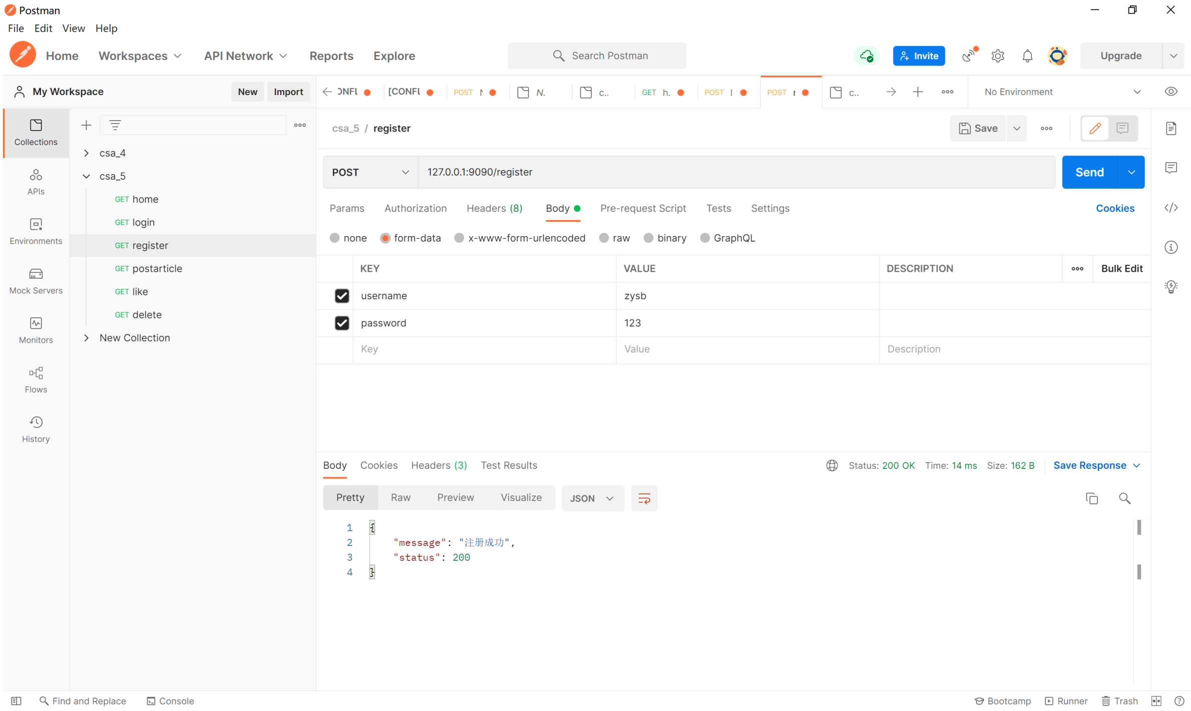
POST 127.0.0.1:9090/register

Body: form-data

	KEY	VALUE	DESCRIPTION
☑	username	zy	
☑	password	123	

Status: 403 Forbidden Time: 3 ms Size: 178 B

```
{
    "message": "已存在该用户名",
    "status": 403
}
```


csa_5 / register

POST 127.0.0.1:9090/register

Body: form-data

	KEY	VALUE	DESCRIPTION
☑	username	zysb	
☑	password	123	

Status: 200 OK Time: 14 ms Size: 162 B

```
{
    "message": "注册成功",
    "status": 200
}
```

2.登录 127.0.0.1:9090/login post请求

```
r.POST("/login", controller.Login)
```

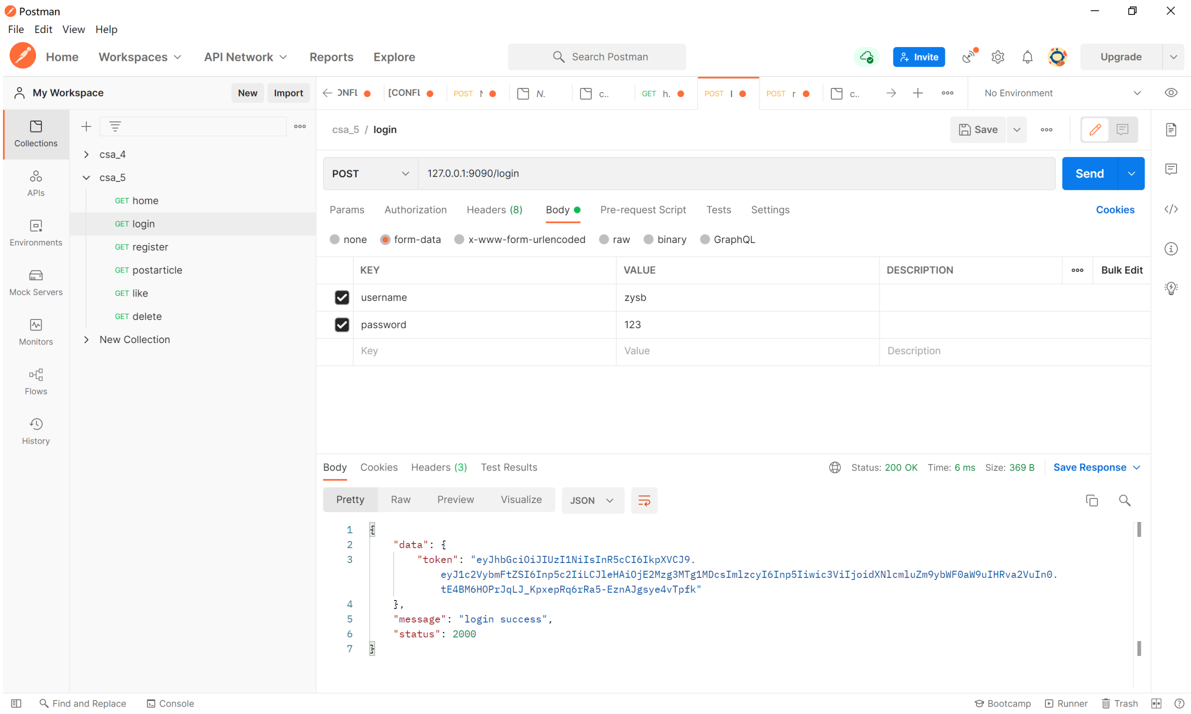
```
//Login
//@title       Login()
//@description 登录请求
//@author      zy
//@param       c *gin.Context
//@return
func Login(c *gin.Context) {
   var u userinformation.UserInfo
   err := c.ShouldBind(&u)              //参数绑定
   if err != nil {
```

```go
        c.JSON(http.StatusOK, gin.H{
            "status": 2001,
            "message": "无效的参数",
        })
        return
    }
    // 判断用户名密码是否正确
    if !common.QueryUserInfoExist(u) {
        c.JSON(http.StatusForbidden, gin.H{
            "status": http.StatusForbidden,
            "message": "用户名或密码有误!",
        })
        return
    }

    // 生成username对应的tokenString
    tokenString, err := common.GenToken(u.Username)
    if err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{
            "status": http.StatusInternalServerError,
            "message": "系统异常",
        })
        return
    }

    c.JSON(http.StatusOK, gin.H{
        "status": 2000,
        "message": "login success",
        "data": gin.H{"token": tokenString},
    })
}
```

测试图：


Postman
File Edit View Help
Home Workspaces API Network Reports Explore
Search Postman
Invite Upgrade
My Workspace New Import
Collections APIs Environments Mock Servers Monitors Flows History
csa_4
csa_5
GET home
GET login
GET register
GET postarticle
GET like
GET delete
New Collection
No Environment
csa_5 / login
Save
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies
Pretty Raw Preview Visualize JSON
Find and Replace Console Bootcamp Runner Trash

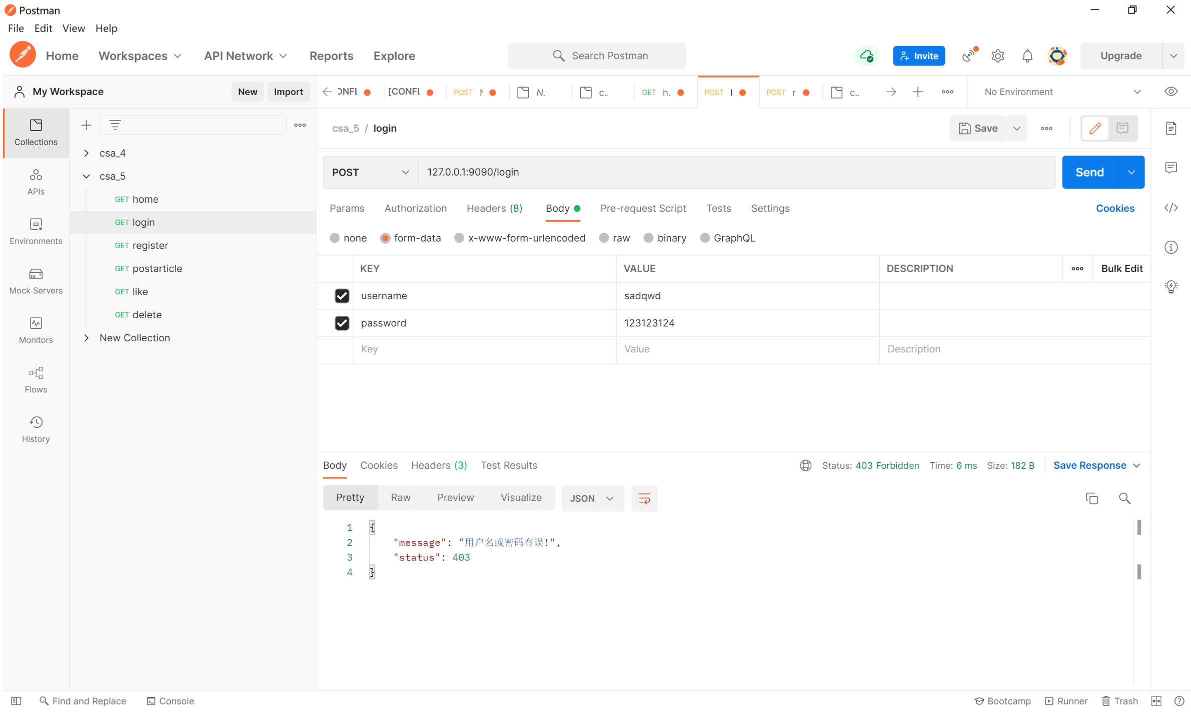
POST 127.0.0.1:9090/login

○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	KEY	VALUE	DESCRIPTION
☑	username	zysb	
☑	password	123	
	Key	Value	Description

Body Cookies Headers (3) Test Results — Status: 200 OK Time: 6 ms Size: 369 B Save Response

```json
{
    "data": {
        "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
            eyJ1c2VybmFtZSI6Inp5c2IiLCJleHAiOjE2Mzg3MTg1MDcsImlzcyI6Inp5Iiwic3ViIjoidXNlcmluZm9ybWF0aW9uIHRva2VuIn0.
            tE4BM6HOPrJqLJ_KpxepRq6rRa5-EznAJgsye4vTpfk"
    },
    "message": "login success",
    "status": 2000
}
```



POST 127.0.0.1:9090/login

○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	KEY	VALUE	DESCRIPTION
☑	username	sadqwd	
☑	password	123123124	
	Key	Value	Description

Body Cookies Headers (3) Test Results

Status: 403 Forbidden Time: 6 ms Size: 182 B

Pretty Raw Preview Visualize JSON

```
{
    "message": "用户名或密码有误!",
    "status": 403
}
```

3.发表文章 127.0.0.1:9090/postArticle post请求

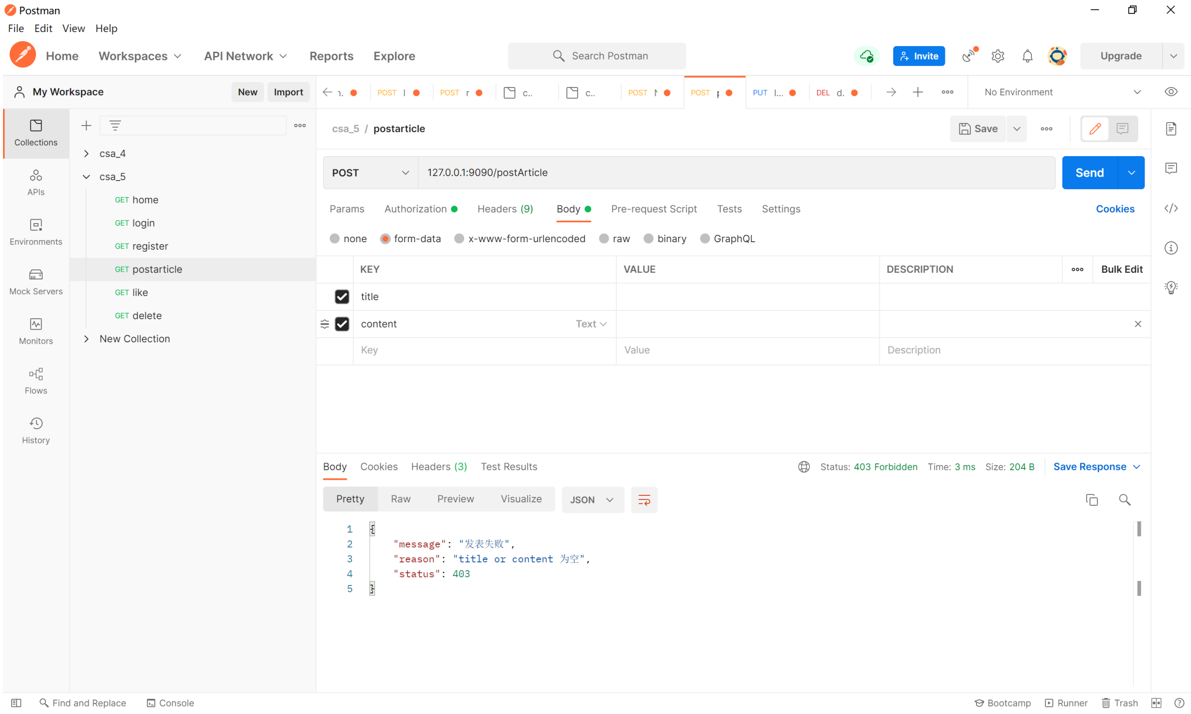
```
r.POST("/postArticle", middleware.JWTAuthMiddleware(), controller.PostArticle)
```

```
//PostArticle
//@title      PostArticle()
//@description 发布文章请求
//@author     zy
//@param      c *gin.Context
//@return
func PostArticle(c *gin.Context) {
    var ArticleInfo userinformation.Article
    username, _ := c.Get("username")      //获取当前登录的username
    ArticleInfo.Username = username.(string)
    err := c.ShouldBind(&ArticleInfo)        //参数绑定

    if err != nil {
        c.JSON(http.StatusOK, gin.H{
            "status": 2001,
            "message": "无效的参数",
        })
        return
    }

    //不能发布空的文章
    if ArticleInfo.Content == "" || ArticleInfo.Title == ""{            //内容或标
题为空      提示错误
        c.JSON(http.StatusForbidden, gin.H{
            "status": http.StatusForbidden,
            "message": "发表失败",
            "reason": "title or content 为空",
        })
        return
    }
    // 是否成功发表文章
    if common.InsertArticle(ArticleInfo) {
        c.JSON(http.StatusOK, gin.H{
            "status": 200,
            "message": "发表成功",
            "data": gin.H{"article": ArticleInfo},
        })
    } else {
        c.JSON(http.StatusBadRequest, gin.H{
            "status": http.StatusBadRequest,
            "message": "发表失败！",
        })
    }
}
```

测试图：



csa_5 / postarticle

POST 127.0.0.1:9090/postArticle

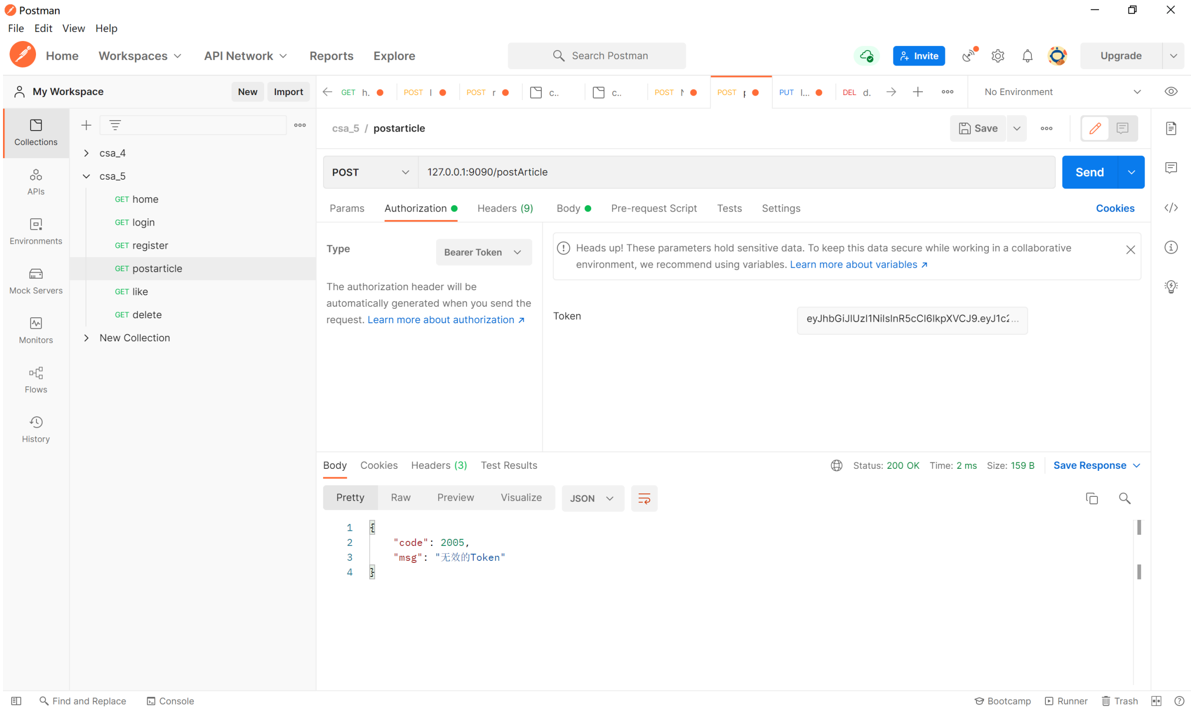
○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	KEY	VALUE	DESCRIPTION
☑	title		
☑	content Text		
	Key	Value	Description

Body Cookies Headers (3) Test Results — Status: 403 Forbidden Time: 3 ms Size: 204 B Save Response

Pretty Raw Preview Visualize JSON

```
{
    "message": "发表失败",
    "reason": "title or content 为空",
    "status": 403
}
```



csa_5 / postarticle

POST 127.0.0.1:9090/postArticle

Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

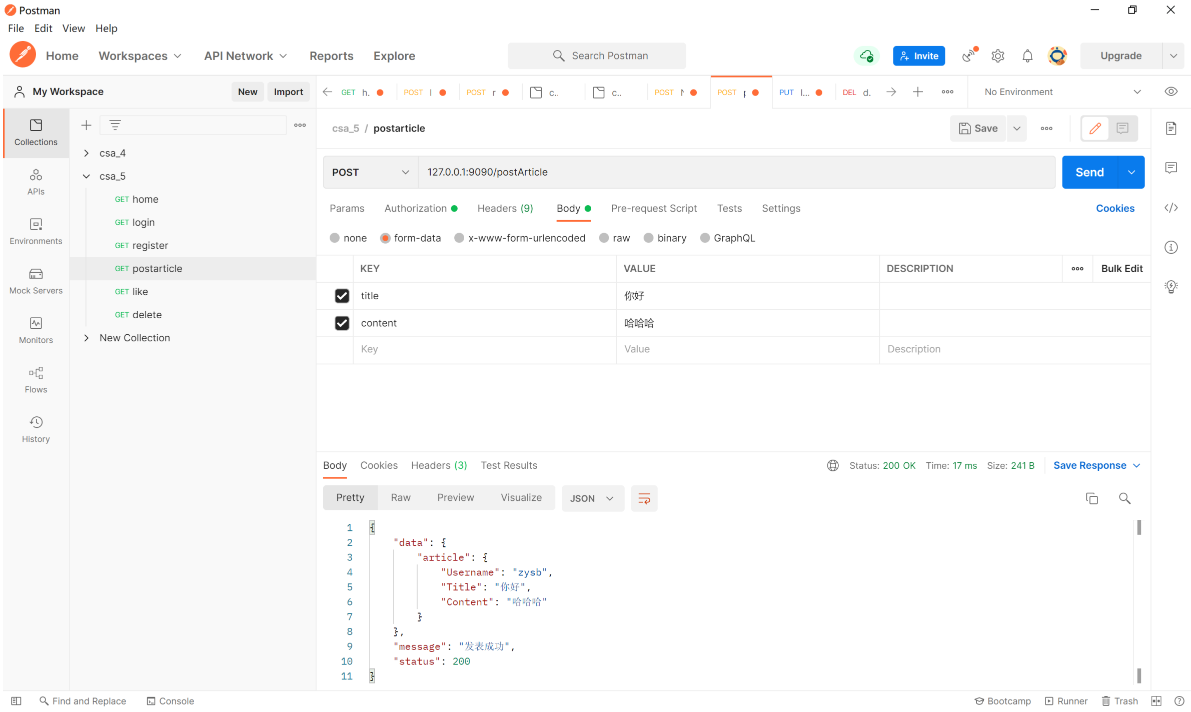
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables ↗

Token: eyJhbGiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c...

Body Cookies Headers (3) Test Results — Status: 200 OK Time: 2 ms Size: 159 B Save Response

Pretty Raw Preview Visualize JSON

```
{
    "code": 2005,
    "msg": "无效的Token"
}
```



csa_5 / postarticle

POST 127.0.0.1:9090/postArticle

Body: form-data

KEY	VALUE	DESCRIPTION
☑ title	你好	
☑ content	哈哈哈	

Status: 200 OK Time: 17 ms Size: 241 B

```
{
    "data": {
        "article": {
            "Username": "zysb",
            "Title": "你好",
            "Content": "哈哈哈"
        }
    },
    "message": "发表成功",
    "status": 200
}
```

4.为文章点赞 127.0.0.1:9090/like put请求

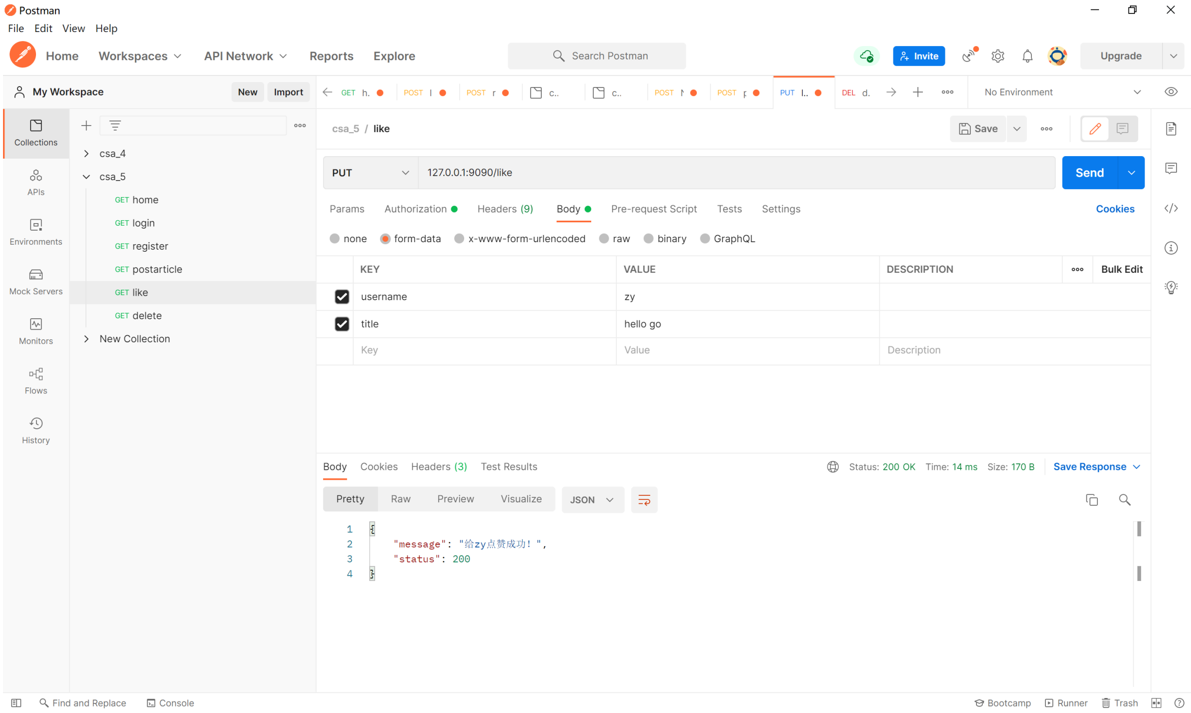
```
r.PUT("/like", middleware.JWTAuthMiddleware(), controller.Like)
```

```
//Like
//@title      Like()
//@description 点赞文章请求
//@author     zy
//@param      c *gin.Context
//@return
func Like(c *gin.Context)  {
   var ArticleInfo userinformation.Article
   username, _ := c.Get("username")  //username为当前用户id
   err := c.ShouldBind(&ArticleInfo)

   if err != nil {
      c.JSON(http.StatusOK, gin.H{
         "status": 2001,
         "message": "无效的参数",
      })
      return
   }

   if username == ArticleInfo.Username {
      c.JSON(http.StatusOK, gin.H{
         "status": http.StatusOK,
         "message": "你不能给自己点赞！",
      })
      return
   }
   n := common.LikeArticle(ArticleInfo)
   if n == 1 {
      c.JSON(http.StatusInternalServerError, gin.H{
         "status": http.StatusInternalServerError,
            "message": "Some errors in sql",
        })
        return
    } else if n == 2 {
        c.JSON(http.StatusForbidden, gin.H{
            "status": http.StatusForbidden,
            "message": "点赞失败，没有相应的文章",
        })
        return
    } else {
        c.JSON(http.StatusOK, gin.H{
            "status": http.StatusOK,
            "message": "给" + ArticleInfo.Username + "点赞成功！",
        })
    }
}
```

测试图：



Postman
File Edit View Help
Home Workspaces API Network Reports Explore
Search Postman
Invite Upgrade
My Workspace New Import
Collections APIs Environments Mock Servers Monitors Flows History
csa_4
csa_5
GET home
GET login
GET register
GET postarticle
GET like
GET delete
New Collection
No Environment
csa_5 / like
Save
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Cookies
Body Cookies Headers (3) Test Results
Save Response
Pretty Raw Preview Visualize JSON
Find and Replace Console Bootcamp Runner Trash

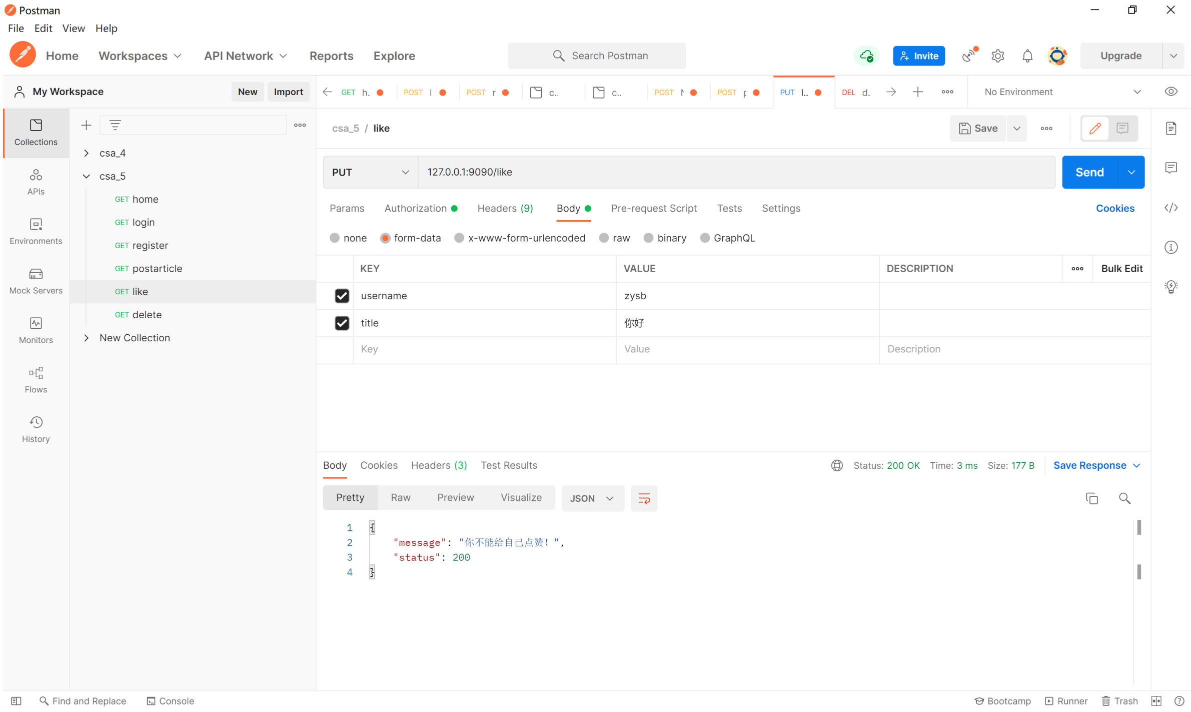
PUT 127.0.0.1:9090/like

○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	KEY	VALUE	DESCRIPTION
☑	username	zy	
☑	title	hello go	
	Key	Value	Description

Status: 200 OK Time: 14 ms Size: 170 B

```
{
    "message": "给zy点赞成功！",
    "status": 200
}
```



PUT 127.0.0.1:9090/like

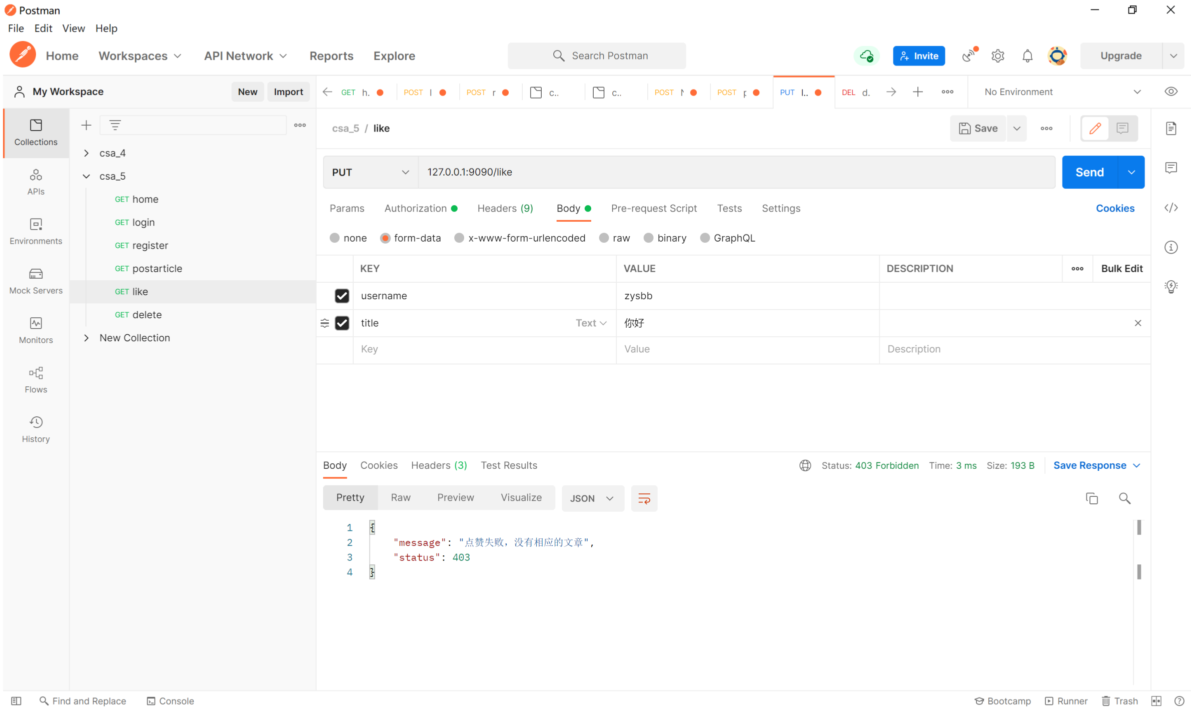
○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	KEY	VALUE	DESCRIPTION
☑	username	zysb	
☑	title	你好	
	Key	Value	Description

Body Cookies Headers (3) Test Results   Status: 200 OK Time: 3 ms Size: 177 B

Pretty Raw Preview Visualize JSON

```
{
    "message": "你不能给自己点赞！",
    "status": 200
}
```



PUT 127.0.0.1:9090/like

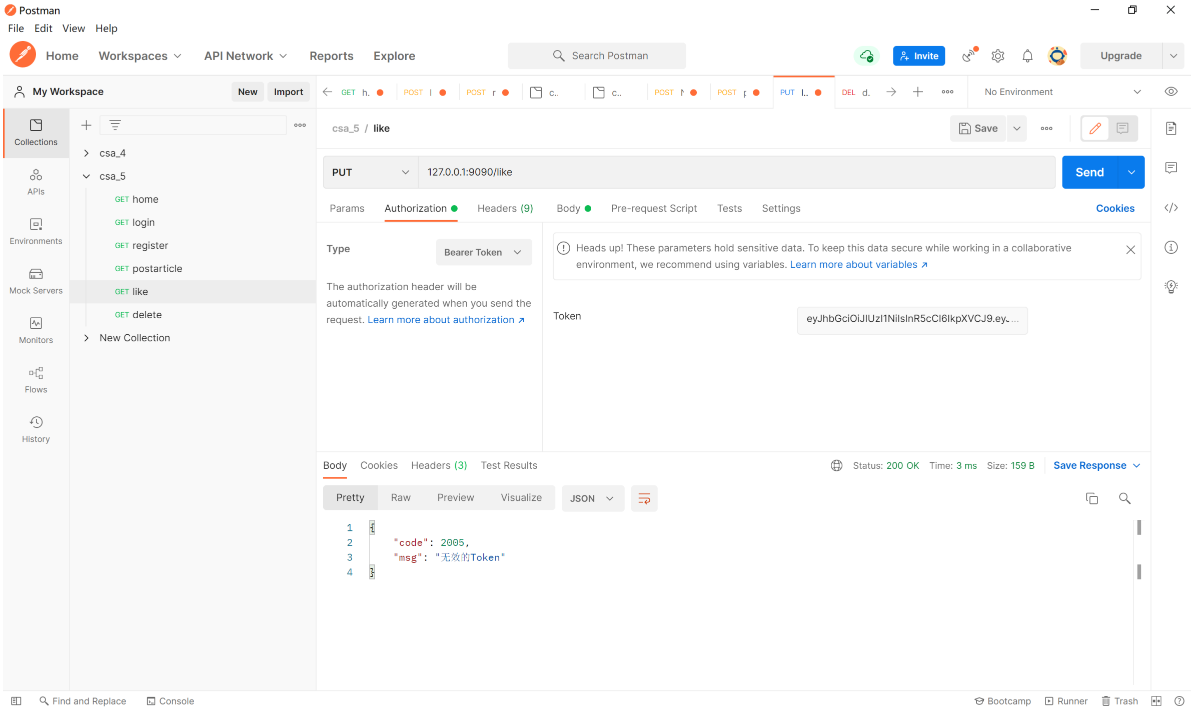
○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	KEY	VALUE	DESCRIPTION
☑	username	zysbb	
☑	title Text	你好	
	Key	Value	Description

Body Cookies Headers (3) Test Results   Status: 403 Forbidden Time: 3 ms Size: 193 B

Pretty Raw Preview Visualize JSON

```
{
    "message": "点赞失败，没有相应的文章",
    "status": 403
}
```



csa_5 / like

PUT 127.0.0.1:9090/like

Type Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables ↗

Token eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey...

Status: 200 OK Time: 3 ms Size: 159 B

```
{
    "code": 2005,
    "msg": "无效的Token"
}
```

5.删除文章 127.0.0.1:9090/deleteArticle delete请求

```
r.DELETE("/deleteArticle", middleware.JWTAuthMiddleware(), controller.Delete)
```

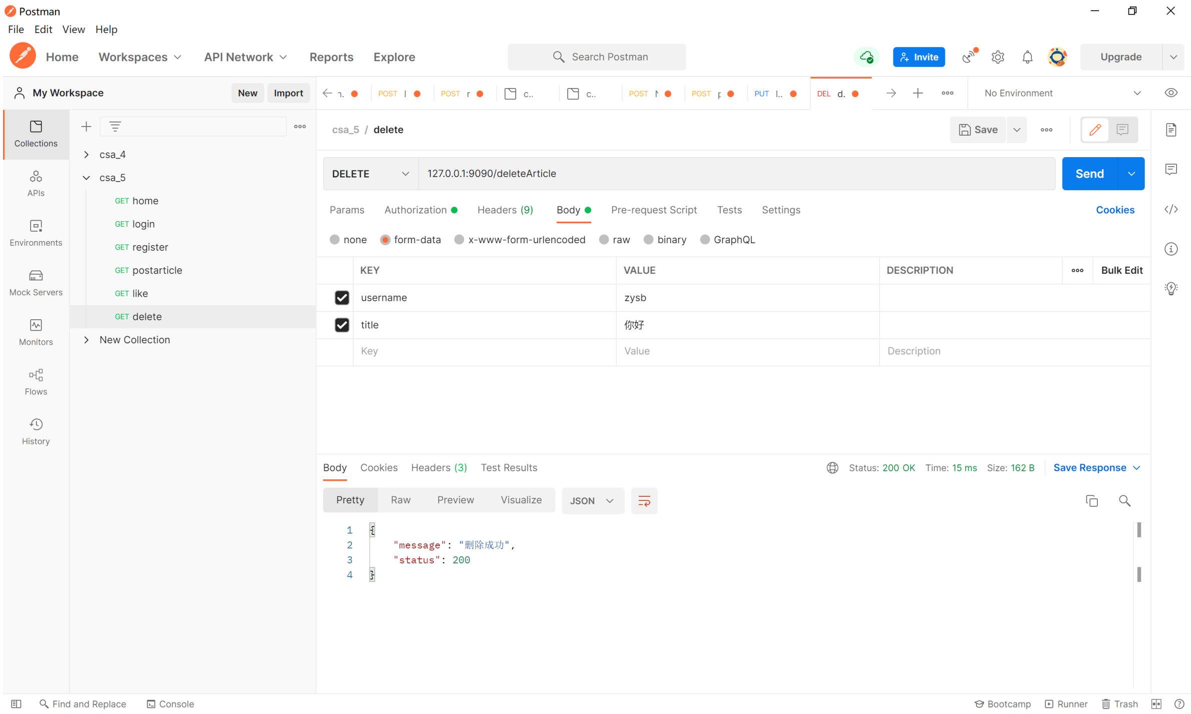
```
//Delete
//@title      Delete()
//@description 删除文章请求
//@author     zy
//@param      c *gin.Context
//@return
func Delete(c *gin.Context) {
   var ArticleInfo userinformation.Article
   username, _ := c.Get("username")      //获取当前登录的username
   err := c.ShouldBind(&ArticleInfo)        //参数绑定

   if err != nil {
      c.JSON(http.StatusOK, gin.H{
         "status": 2001,
         "message": "无效的参数",
      })
      return
   }

   if username != ArticleInfo.Username {
      c.JSON(http.StatusForbidden, gin.H{
         "status": http.StatusForbidden,
         "message": "你没有权限删除别人的文章",
      })
      return
   }
   if common.DeleteArticle(ArticleInfo) {
      c.JSON(http.StatusOK, gin.H{
         "status": http.StatusOK,
         "message": "删除成功",
        })
    } else {
        c.JSON(http.StatusForbidden, gin.H{
            "status": http.StatusForbidden,
            "message": "删除失败",
        })
    }

}
```

测试图：



csa_5 / delete

DELETE 127.0.0.1:9090/deleteArticle

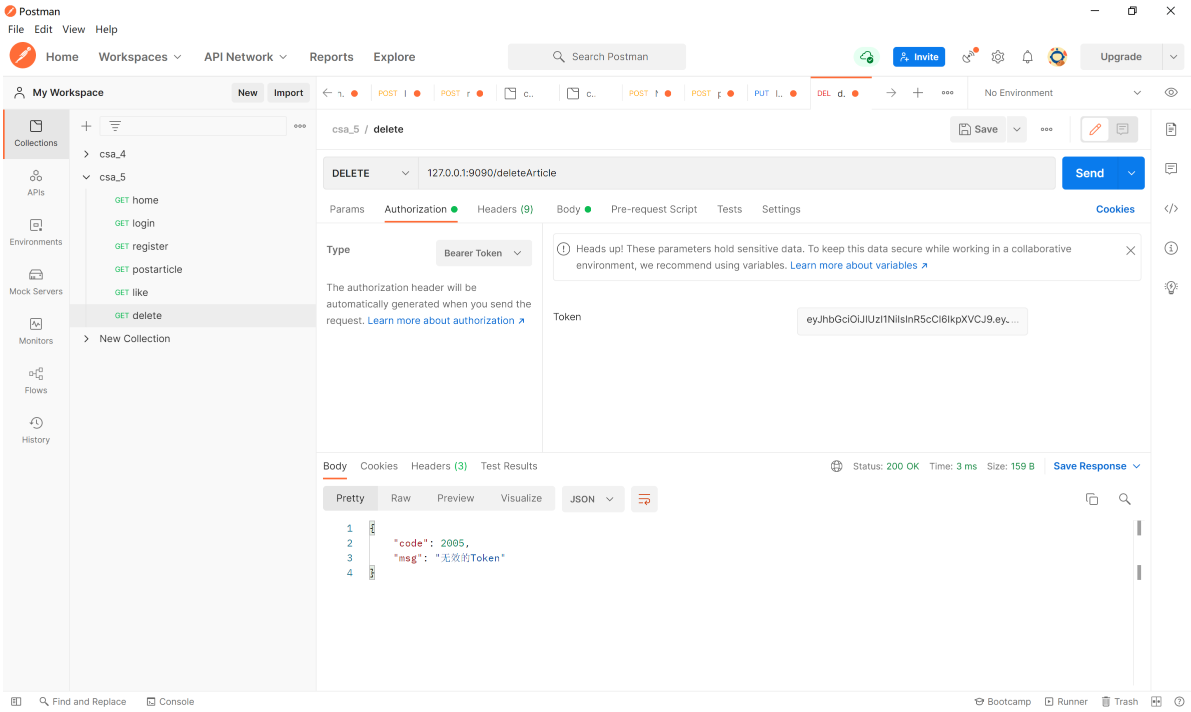
Body: form-data

KEY	VALUE	DESCRIPTION
☑ username	zysb	
☑ title	你好	

Status: 200 OK Time: 15 ms Size: 162 B

```
{
    "message": "删除成功",
    "status": 200
}
```

Postman
File Edit View Help
Home Workspaces API Network Reports Explore
Search Postman
Invite Upgrade
My Workspace New Import
No Environment
Collections APIs Environments Mock Servers Monitors Flows History
csa_4
csa_5
GET home
GET login
GET register
GET postarticle
GET like
GET delete
New Collection
Save Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies
none form-data x-www-form-urlencoded raw binary GraphQL
Bulk Edit
Body Cookies Headers (3) Test Results Save Response
Pretty Raw Preview Visualize JSON
Find and Replace Console Bootcamp Runner Trash



csa_5 / delete

DELETE 127.0.0.1:9090/deleteArticle

Authorization

Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

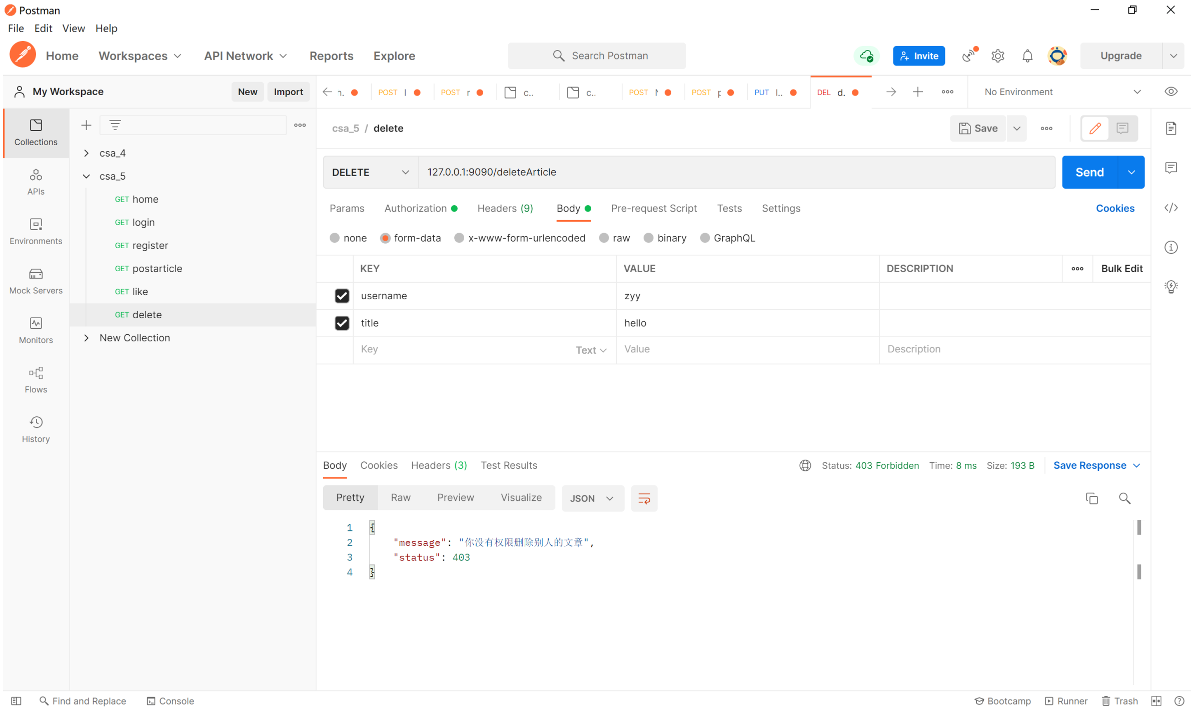
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables ↗

Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey...

Status: 200 OK Time: 3 ms Size: 159 B

```
{
    "code": 2005,
    "msg": "无效的Token"
}
```

Postman
File Edit View Help
Home Workspaces API Network Reports Explore
Search Postman
Invite Upgrade
My Workspace New Import
No Environment
Collections APIs Environments Mock Servers Monitors Flows History
csa_4
csa_5
GET home
GET login
GET register
GET postarticle
GET like
GET delete
New Collection
Save Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies
Body Cookies Headers (3) Test Results Save Response
Pretty Raw Preview Visualize JSON
Find and Replace Console Bootcamp Runner Trash



csa_5 / delete

DELETE 127.0.0.1:9090/deleteArticle

Body: form-data

KEY	VALUE	DESCRIPTION
☑ username	zyy	
☑ title	hello	

Status: 403 Forbidden Time: 8 ms Size: 193 B

```
{
    "message": "你没有权限删除别人的文章",
    "status": 403
}
```

6.留言 127.0.0.1:9090/message post请求

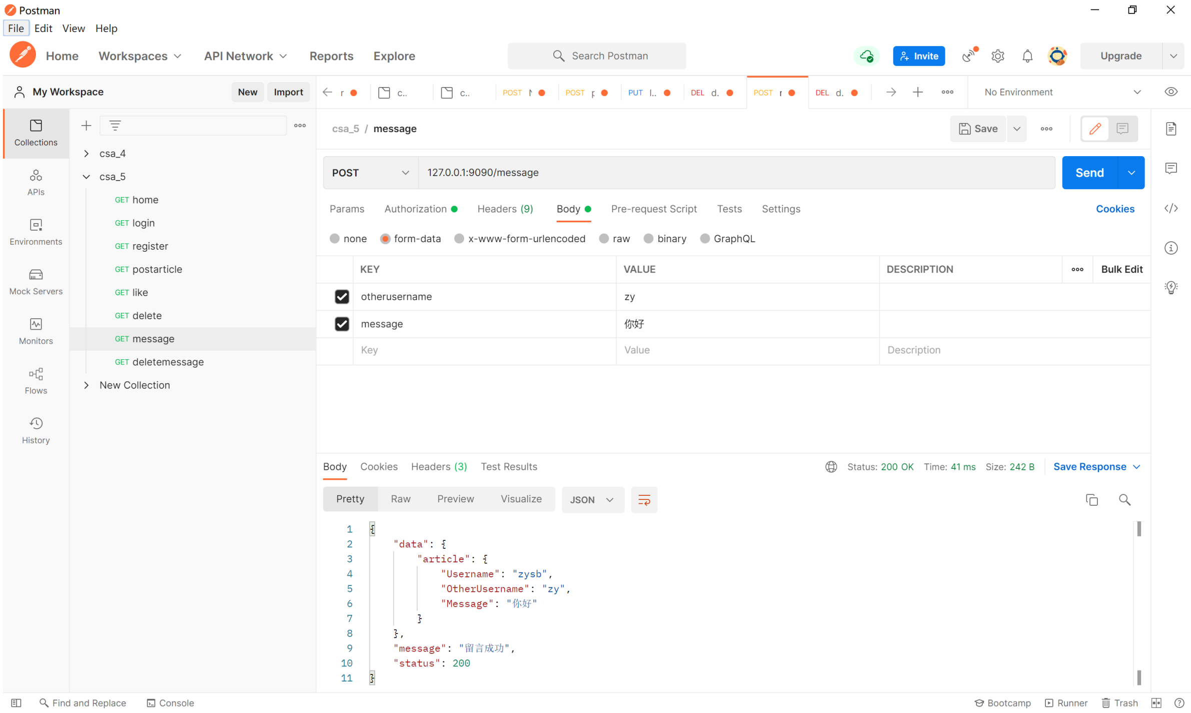
```
r.POST("/message", middleware.JWTAuthMiddleware(), controller.MessageToOther)
```

```
//MessageToOther
//@title       MessageToOther()
//@description 给其他用户留言请求
//@author      zy
//@param       c *gin.Context
//@return
func MessageToOther(c *gin.Context) {
   var MsgTo userinformation.Msg
   username, _ := c.Get("username")  //username为当前用户id
   err := c.ShouldBind(&MsgTo)
   MsgTo.Username = username.(string)
   if err != nil {
      c.JSON(http.StatusOK, gin.H{
         "status": 2001,
         "message": "无效的参数",
      })
      return
   }

   if MsgTo.OtherUsername == "" || MsgTo.Message == "" {
      c.JSON(http.StatusForbidden, gin.H{
         "status": http.StatusForbidden,
         "message": "留言失败",
         "reason": "目标username或留言内容为空",
      })
      return
   }

   if common.MessageInsert(MsgTo) {
      c.JSON(http.StatusOK, gin.H{
            "status": 200,
            "message": "留言成功",
            "data": gin.H{"article": MsgTo},
        })
        return
    } else {
        c.JSON(http.StatusBadRequest, gin.H{
            "status": http.StatusBadRequest,
            "message": "留言失败！",
        })
    }
}
```

测试图：



csa_5 / message

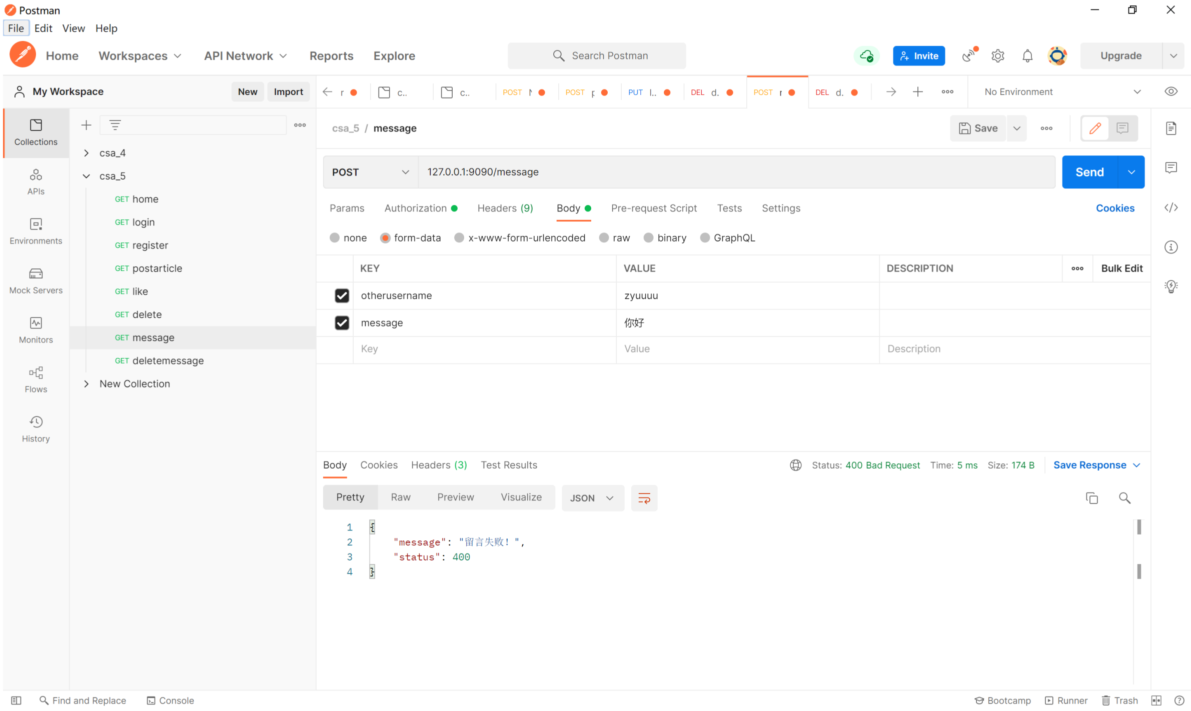
POST 127.0.0.1:9090/message

Body: form-data

KEY	VALUE	DESCRIPTION
☑ otherusername	zy	
☑ message	你好	

Status: 200 OK Time: 41 ms Size: 242 B

```
{
    "data": {
        "article": {
            "Username": "zysb",
            "OtherUsername": "zy",
            "Message": "你好"
        }
    },
    "message": "留言成功",
    "status": 200
}
```



csa_5 / message

POST 127.0.0.1:9090/message

Body: form-data

KEY	VALUE	DESCRIPTION
☑ otherusername	zyuuuu	
☑ message	你好	

Status: 400 Bad Request Time: 5 ms Size: 174 B

```
{
    "message": "留言失败！",
    "status": 400
}
```

7.删除留言 127.0.0.1:9090/deleteMessage delete请求

```
r.DELETE("/deleteMessage", middleware.JWTAuthMiddleware(), controller.DeleteMsg)
```

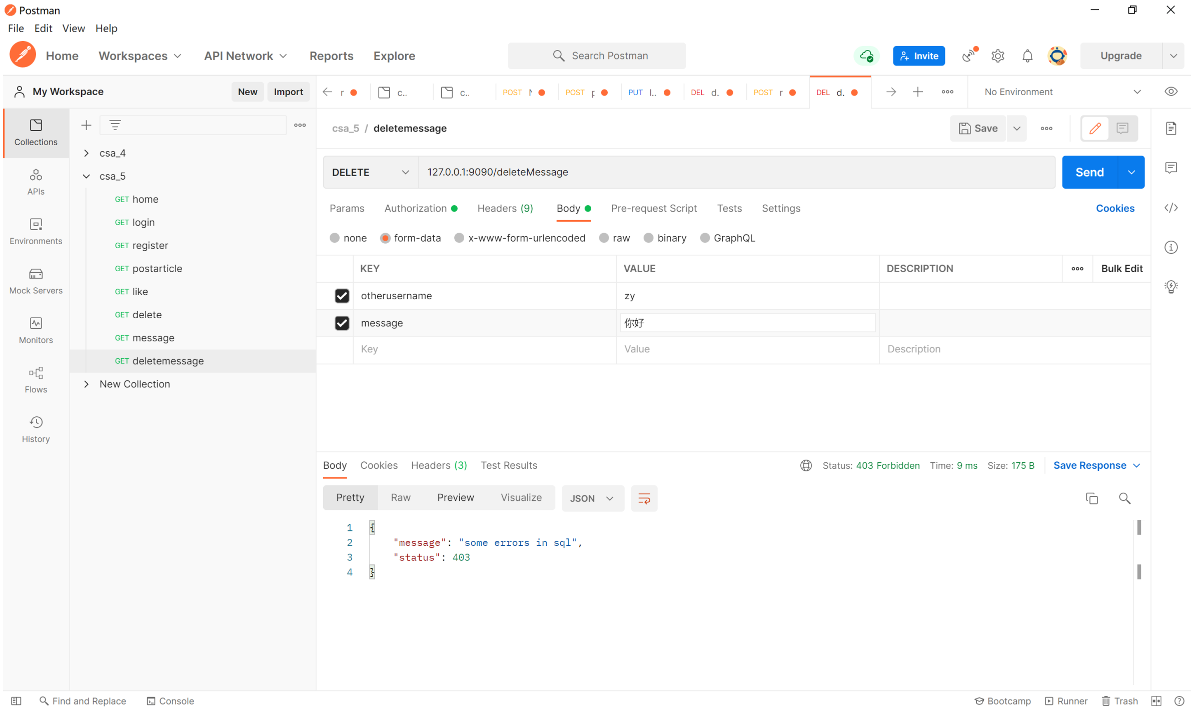
```
//DeleteMsg
//@title      DeleteMsg()
//@description 删除留言请求
//@author     zy
//@param      c *gin.Context
//@return
func DeleteMsg(c *gin.Context) {
   var MsgTo userinformation.Msg
   username, _ := c.Get("username")  //username为当前用户id
   err := c.ShouldBind(&MsgTo)
   MsgTo.Username = username.(string)
   if err != nil {
      c.JSON(http.StatusOK, gin.H{
         "status": 2001,
         "message": "无效的参数",
      })
      return
   }


   if MsgTo.OtherUsername == "" || MsgTo.Message == "" {
      c.JSON(http.StatusForbidden, gin.H{
         "status": http.StatusForbidden,
         "message": "删除留言失败",
         "reason": "目标username或留言内容为空",
      })
      return
   }

   if !common.MessageDelete(MsgTo) {
      c.JSON(http.StatusForbidden, gin.H{
         "status": http.StatusForbidden,
         "message": "some errors in sql",
      })
      return
   }

   c.JSON(http.StatusOK, gin.H{
      "status": http.StatusOK,
      "message": "删除留言成功",
   })
}
```

测试图：



csa_5 / deletemessage

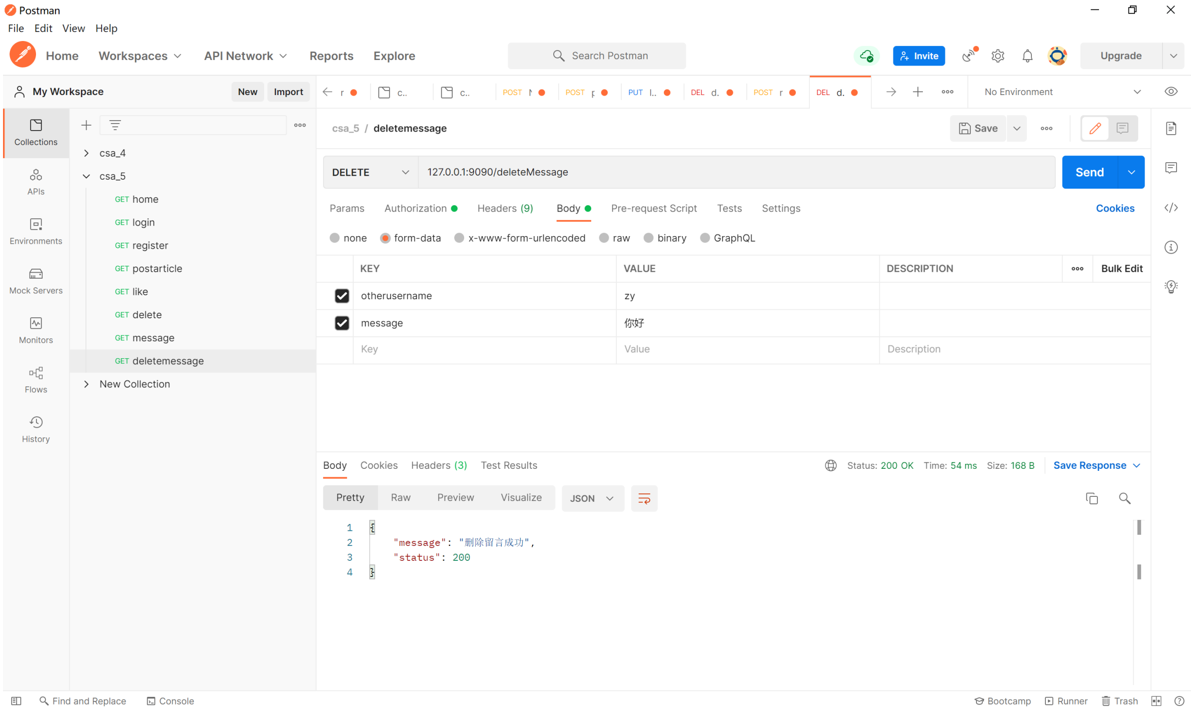
DELETE 127.0.0.1:9090/deleteMessage

○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	KEY	VALUE	DESCRIPTION
☑	otherusername	zy	
☑	message	你好	
	Key	Value	Description

Body Cookies Headers (3) Test Results — Status: 403 Forbidden Time: 9 ms Size: 175 B

```
{
    "message": "some errors in sql",
    "status": 403
}
```



csa_5 / deletemessage

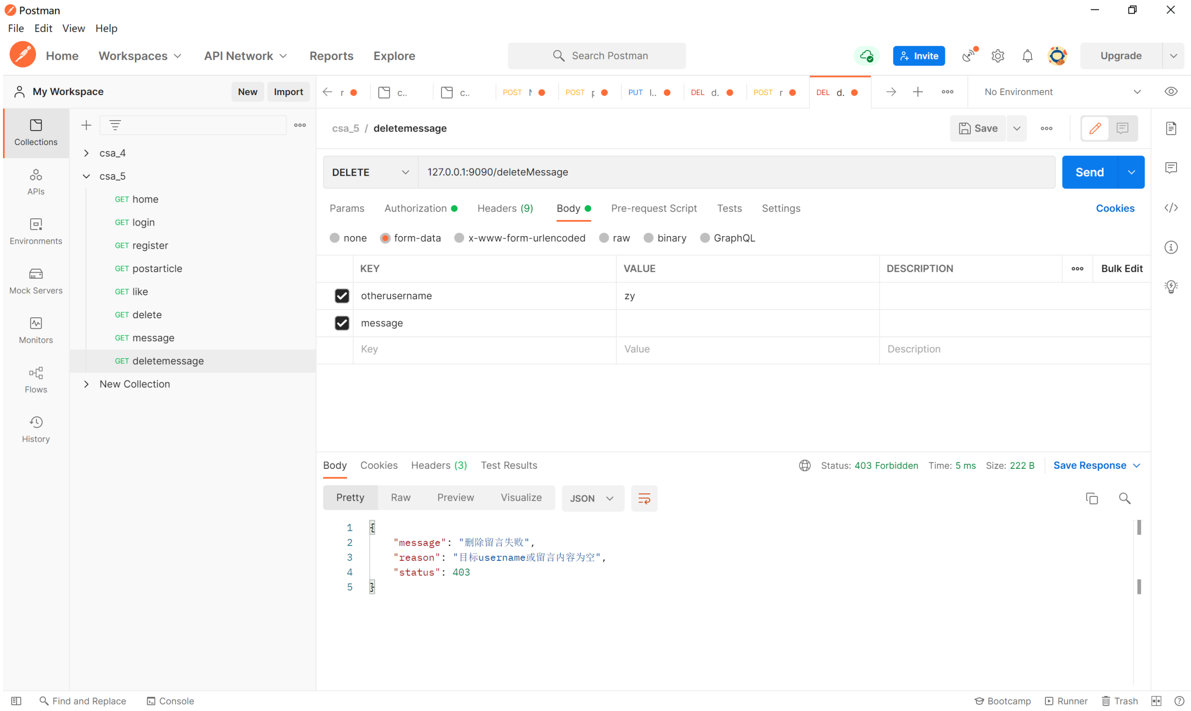
DELETE 127.0.0.1:9090/deleteMessage

○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	KEY	VALUE	DESCRIPTION
☑	otherusername	zy	
☑	message	你好	
	Key	Value	Description

Body Cookies Headers (3) Test Results — Status: 200 OK Time: 54 ms Size: 168 B

```
{
    "message": "删除留言成功",
    "status": 200
}
```


Postman
File Edit View Help
Home Workspaces API Network Reports Explore
Search Postman
Invite Upgrade
My Workspace New Import
Collections APIs Environments Mock Servers Monitors Flows History
csa_4
csa_5
GET home
GET login
GET register
GET postarticle
GET like
GET delete
GET message
GET deletemessage
New Collection
No Environment
Save
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies
Find and Replace Console Bootcamp Runner Trash

csa_5 / deletemessage

DELETE 127.0.0.1:9090/deleteMessage

Body: ○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	KEY	VALUE	DESCRIPTION
☑	otherusername	zy	
☑	message		
	Key	Value	Description

Body Cookies Headers (3) Test Results — Status: 403 Forbidden Time: 5 ms Size: 222 B Save Response

Pretty Raw Preview Visualize JSON

```
{
    "message": "删除留言失败",
    "reason": "目标username或留言内容为空",
    "status": 403
}
```

数据库相关操作

```
//@Title       db.go
//@Description 数据库相关操作
//@Author      zy
//@Update      2021.12.5

package common

import (
    "csa_5/userinformation"
    "database/sql"
    "fmt"
    _ "github.com/go-sql-driver/mysql"
)

// DB 定义一个全局变量
var DB *sql.DB

//InitDB
//@title       InitDB()
//@description 连接数据库
//@author      zy
//@param       dsn string
//@return      *sql.DB error
func InitDB(dsn string) (*sql.DB, error) {
    var err error
    DB, err = sql.Open("mysql", dsn)
    if err != nil {
        fmt.Printf("failed to open database, err:%v", err)
        return nil, err
    }
    err = DB.Ping()
```

```go
    if err != nil {
        fmt.Printf("failed to connect database, err: %v", err)
        return nil, err
    }
    return DB, err
}

//QueryUserInfo
//@title      QueryUserInfo()
//@description 查询用户名u.Username是否已经存在
//@author     zy
//@param      u userinformation.UserInfo
//@return     bool
func QueryUserInfo(u userinformation.UserInfo) bool {
    sqlStr := "select username from user where username=?"       //sql语句
    var UTemp userinformation.UserInfo
    err := DB.QueryRow(sqlStr, u.Username).Scan(&UTemp.Username)   //调用QueryRow
进行插入
    if err != nil {
        fmt.Printf("用户名%s还未注册\n", u.Username)
        return false
    }
    fmt.Printf("用户:%s你好\n", UTemp.Username)
    return true
}

//QueryUserInfoExist
//@title      QueryUserInfoExist()
//@description 查询用户名和密码是否正确
//@author     zy
//@param      u userinformation.UserInfo
//@return     bool
func QueryUserInfoExist(u userinformation.UserInfo) bool {
    sqlStr := "select username, password from user where username=? and
password=?"    //sql语句
    var UTemp userinformation.UserInfo

    err := DB.QueryRow(sqlStr, u.Username, u.Password).Scan(&UTemp.Username,
&UTemp.Password)  //调用QueryRow进行插入

    if err != nil {
        fmt.Printf("scan failed, err:%v\n", err)
        return false
    }
    fmt.Printf("用户:%s你好\n", UTemp.Username)
    return true
}

//InsertUserInfo
//@title      InsertUserInfo()
//@description 注册成功时插入到user表中
//@author     zy
//@param      u userinformation.UserInfo
//@return     bool
func InsertUserInfo(u userinformation.UserInfo) bool{
    sqlStr := "insert into user(username, password) values (?,?)"  //sql语句
    ret, err := DB.Exec(sqlStr, u.Username, u.Password)             //插入操作
    if err != nil {
        fmt.Printf("insert failed, err:%v", err)
        return false
    }
    id, err := ret.LastInsertId()
    if err != nil {
        fmt.Printf("get lastinsert ID failed, err: %v\n", err)
        return false
    }
    fmt.Printf("insert success, the id is %d.\n", id)
    return true
}

//InsertArticle
//@title       InsertArticle()
//@description 成功发表文章时插入到blog表中
//@author      zy
//@param       uArticle userinformation.Article
//@return      bool
func InsertArticle(uArticle userinformation.Article) bool{
    sqlStr := "insert into blog(username, title, content) values (?,?,?)"
 //sql语句
    ret, err :=DB.Exec(sqlStr, uArticle.Username, uArticle.Title,
uArticle.Content)    //sql操作
    if err != nil {
        fmt.Printf("insert failed, err:%v\n", err)
        return false
    }
    id, err := ret.LastInsertId()
    if err != nil {
        fmt.Printf("get lastinsert ID failed, err: %v\n", err)
        return false
    }
    fmt.Printf("insert success, the id is %d.\n", id)
    return true
}

//DeleteArticle
//@title       DeleteArticle()
//@description 成功删除文章时将blog表中对用的数据删除
//@author      zy
//@param       uArticle userinformation.Article
//@return      bool
func DeleteArticle(uArticle userinformation.Article) bool{
    sqlStr := "delete from blog where username=? and title=?"        //sql语句
    ret, err := DB.Exec(sqlStr, uArticle.Username, uArticle.Title)    //sql操作
    if err != nil {
        fmt.Printf("delete fail, err: %v\n", err)
        return false
    }
    n, err := ret.RowsAffected()
    if err != nil {
        fmt.Printf("affect fail, err: %v\n", err)
        return false
    }
    fmt.Printf("delete success, delete %d article which title is %s", n,
uArticle.Title)
    return true
```

```
}

//LikeArticle
//@title       LikeArticle()
//@description 点赞别人的文章
//@author      zy
//@param       uArticle userinformation.Article
//@return      bool
func LikeArticle(uArticle userinformation.Article) int{
   sqlStr := "update blog set favor=favor+1 where username=? and title = ?"
    //sql语句
   ret, err := DB.Exec(sqlStr, uArticle.Username, uArticle.Title)
             //更新对应的值--favor++
   if err != nil {                                                       //存在错
误  提示用户
      fmt.Printf("failed to update, err:%v", err)
      return 1
   }
   n, err1 := ret.RowsAffected()                                         //判断更
新了几行相应的值  即判断username和title是否存在
   if err1 != nil {
      fmt.Printf("failed to affect, err:%v", err)
      return 1
   }
   if n == 0 {                                                          //没有相应
的博客
      fmt.Printf("faied to like because %s doesn't write %s , err:%v",
uArticle.Username, uArticle.Title, err)
      return 2
   }

   fmt.Printf("success like")
   return 3
}

//MessageInsert
//@title       MessageInsert()
//@description 留言成功时插入msg表
//@author      zy
//@param       Msg userinformation.Msg
//@return      bool
func MessageInsert(Msg userinformation.Msg) bool{
   // 判断是否有Msg.OtherUsername 这个用户
   if !QueryUserInfo(userinformation.UserInfo{Username: Msg.OtherUsername}) {
      fmt.Printf("没有这个用户")
      return false
   }
   sqlStr := "insert into msg(username, otherusername, message) values (?,?,?)"
        //sql语句
   ret, err := DB.Exec(sqlStr, Msg.Username, Msg.OtherUsername, Msg.Message)
      //插入操作
   if err != nil {
      fmt.Printf("insert failed, err:%v", err)
      return false
   }
   id, err := ret.LastInsertId()
   if err != nil {
      fmt.Printf("get lastinsert ID failed, err: %v\n", err)
```

```
        return false
    }
    fmt.Printf("insert success, the id is %d.\n", id)
    return true
}

//MessageDelete
//@title      MessageDelete()
//@description 删除留言
//@author     zy
//@param      Msg userinformation.Msg
//@return     bool
func MessageDelete(Msg userinformation.Msg) bool {
    sqlStr := "delete from msg where username=? and otherusername=? and
message=?"       //sql语句
    ret, err := DB.Exec(sqlStr, Msg.Username, Msg.OtherUsername, Msg.Message)
//sql操作
    if err != nil {
        fmt.Printf("delete fail, err: %v\n", err)
        return false
    }
    n, err := ret.RowsAffected()
    if err != nil {
        fmt.Printf("affect fail, err: %v\n", err)
        return false
    }
    fmt.Printf("delete success, delete %d message which username is %s and
otherusername is %s and message is %s\n", n, Msg.Username, Msg.OtherUsername,
Msg.Message)
    return true
}
```

JWT鉴权及中间件

```
//@Title       jwt.go
//@Description jwt鉴权
//@Author     zy
//@Update     2021.12.5

package common

import (
    "errors"
    "github.com/dgrijalva/jwt-go"
    "time"
)

// MyClaims 结构体
type MyClaims struct {
    Username string `json:"username"`
    jwt.StandardClaims
}
```

```
// JWT过期时间
const TokenExpireDuration = time.Hour * 2

// Secret签名
var MySecret = []byte("zyaichirou")

//GenToken
//@title      GenToken()
//@description 生成JWT
//@author     zy
//@param      username string
//@return     string error
func GenToken(username string) (string, error) {
    c := MyClaims{
        username,
        jwt.StandardClaims{
            ExpiresAt: time.Now().Add(TokenExpireDuration).Unix(),   //过期时间
            Issuer: "zy",                                   //签发人
            Subject: "userinformation token",                     //签发主题
        },
    }

    // 使用指定的签名方法创建签名对象
    token := jwt.NewWithClaims(jwt.SigningMethodHS256, c)
    // 使用指定的secret签名
    tokenString, err := token.SignedString(MySecret)
    if err != nil {
        return "", err
    }

    return tokenString, err
}

//ParseToken
//@title      ParseToken()
//@description 解析JWT
//@author     zy
//@param      tokenString string
//@return     *MyClaims error
func ParseToken(tokenString string) (*MyClaims, error) {
    //解析token
    token, err := jwt.ParseWithClaims(tokenString, &MyClaims{}, func(token *jwt.Token) (interface{}, error) {
        return MySecret, nil
    })

    if err != nil {
        return nil, err
    }
    if claims, ok := token.Claims.(*MyClaims); ok && token.Valid {
        return claims, nil
    }  //校验token
    return nil, errors.New("invalid token")
}
```

```
//@Title       AuthMiddleware.go
//@Description jwt鉴权中间件实现
```

```
//@Author     zy
//@Update     2021.12.5

package middleware

import (
   "csa_5/common"
   "csa_5/userinformation"
   "github.com/gin-gonic/gin"
   "net/http"
   "strings"
)


//JWTAuthMiddleware
//@title      JWTAuthMiddleware()
//@description JWT鉴权中间件
//@author     zy
//@param
//@return     gin.HandlerFunc
func JWTAuthMiddleware() gin.HandlerFunc {
   return func(c *gin.Context) {
      authHeader := c.Request.Header.Get("Authorization")       //获取Header的
Authorization
      if authHeader == "" {
         c.JSON(http.StatusOK, gin.H{
            "code": 2003,
            "msg": "请求头中auth为空",
         })
         c.Abort()
         return
      }

      // 按空格分割
      parts := strings.SplitN(authHeader, " ", 2)
      if !(len(parts) == 2 && parts[0] == "Bearer") {
         c.JSON(http.StatusOK, gin.H{
            "code": 2004,
            "msg": "请求头中auth格式有误",
         })
         c.Abort()
         return
      }
      // parts[1]是获取到的tokenString，使用之前解析JWT函数来解析
      mc, err := common.ParseToken(parts[1])
      if err != nil {
         c.JSON(http.StatusOK, gin.H{
            "code": 2005,
            "msg": "无效的Token",
         })
         c.Abort()
         return
      }


      // 在数据库中验证mc中的Username
      // 若不存在则返回
      var u userinformation.UserInfo
```

```
        u.Username = mc.Username
        if !common.QueryUserInfo(u) {
            c.JSON(http.StatusUnprocessableEntity, gin.H{
                "status": 401,
                "message": "权限不足",
            })
            c.Abort()
            return
        }

        //用户存在 将user信息写入请求的上下文c中
        c.Set("username", mc.Username)
        c.Next()   // 后续的处理函数可以用过c.Get("username")来获取当前请求的用户信息
    }
}
```