/v3/api-docs Explore

SpringDoc API Test 0.0.1 OAS3

/v3/api-docs

SpringDoc Simple Application Test

Servers

http://localhost:8080 - Generated server url

ques-controller

GET /questions list questions by page 分页查询 前端传2个参数，一个页码，一个每页的数据数。如果前端没有传送数据 默认为 1 和 10

Parameters Try it out

Name	Description
pageNum integer($int32) *(query)*	*Default value* : 1 1
pageSize integer($int32) *(query)*	*Default value* : 10 10

Responses

Code	Description	Links
200	OK Media type */* Controls Accept header. Example Value Schema	*No links*

```
{
  "code": 0,
  "msg": "string",
  "data": {}
}
```

PUT /questions update the question 更改问题的流程是 前端点击编辑按钮后，发送给后端一个id，后端先根据id找到指定question（即先使用selectById），然后后端会发送question数据给前端，前端修改后发送给后端，再使用这个接口

Parameters Try it out

No parameters

Request body required application/json

Example Value Schema

```
{
  "id": 0,
  "questionId": "string",
  "question": "string",
  "answer1Id": "string",
  "answer1": "string",
  "answer2Id": "string",
  "answer2": "string",
  "answer3Id": "string",
  "answer3": "string",
  "answer4Id": "string",
  "answer4": "string"
}
```

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema

```
{
  "code": 0,
  "msg": "string",
  "data": {}
}
```

POST **/questions** add new question 自动增加在最后 前端传除了id以外的数据值

Parameters

Try it out

No parameters

Request body required

application/json

Example Value Schema

```
{
  "id": 0,
  "questionId": "string",
  "question": "string",
  "answer1Id": "string",
  "answer1": "string",
  "answer2Id": "string",
  "answer2": "string",
  "answer3Id": "string",
  "answer3": "string",
  "answer4Id": "string",
  "answer4": "string"
}
```

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema

```
{
  "code": 0,
  "msg": "string",
```

Code	Description	Links

```
  "data": {}
}
```

GET **/questions/{id}** select question by id 前端传入一个id 后端回传给前端此id对应的question数据

Parameters Try it out

Name	Description
id * required integer($int32) *(path)*	id

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema

```
{
  "code": 0,
  "msg": "string",
  "data": {}
}
```

DELETE **/questions/{id}** delete question by id 前端传过来一个id

Parameters Try it out

Name	Description
id * required integer($int32) *(path)*	id

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema

```
{
  "code": 0,
  "msg": "string",
  "data": {}
}
```

param-controller

GET **/parameters** list parameters by page 分页查询 前端传2个参数，一个页码，一个每页的数据数。如果前端没有传送数据 默认为 1 和 10

Parameters Try it out

Name	Description
pageNum integer($int32) *(query)*	*Default value* : 1 1
pageSize integer($int32) *(query)*	*Default value* : 10 10

Responses

Code	Description	Links
200	OK Media type */* Controls Accept header. Example Value Schema	*No links*

```
{
  "code": 0,
  "msg": "string",
  "data": {}
}
```

PUT **/parameters** update the parameter value 更改parameterValue的流程是 前端点击编辑按钮后，发送给后端一个id，后端先根据id找到指定question（即先使用selectById），然后后端会发送parameter数据给前端，前端修改后发送给后端，再使用这个接口,只能修改parameterValue

Parameters Try it out

No parameters

Request body required application/json

Example Value Schema

```
{
  "id": 0,
  "paramName": "string",
  "paramValue": "string"
}
```

Responses

Code	Description	Links
200	OK Media type */*	*No links*

Code	Description	Links
	Controls `Accept` header. Example Value Schema	

```
{
  "code": 0,
  "msg": "string",
  "data": {}
}
```

GET **/parameters/{id}** select parameter by id 前端传入一个id 后端回传给前端此id对应的parameter数据

Parameters

Try it out

Name	Description
id * required integer($int32) *(path)*	id

Responses

Code	Description	Links
200	OK Media type */* Controls `Accept` header. Example Value Schema	*No links*

```
{
  "code": 0,
  "msg": "string",
  "data": {}
}
```

Schemas

```
Question    {
    id                          [...]
    questionId                  [...]
    question                    [...]
    answer1Id                   [...]
    answer1                     [...]
    answer2Id                   [...]
    answer2                     [...]
    answer3Id                   [...]
    answer3                     [...]
    answer4Id                   [...]
    answer4                     [...]
}
```

```
Result    {
    code                        [...]
    msg                         [...]
    data                        {...}
}
```

```
Parameter   {
   id
                          [...]
   paramName
                          [...]
   paramValue
                          [...]
}
```