

API- Meta Model

版本 2.0

2029/12/12

1、服务版本

API- Meta Model 2.0

2、项目说明

学生信息管理系统的功能是收集学生的个人信息，以便向老师提供每个学生在校或毕业生学籍的情况，还可以让学生用自己的学号去查看自己在校期间的表现。

2-1、学院信息（Department）及其集合（Departments）

描述

可以管理学院信息，实现学院信息的增删改查

属性(数据)

属性	属性描述	是否必填	备注
dep_id	学院编号，主键	新增时必填	最长20位,唯一
dep_name	学院名称	是	最长20位,唯一
master_name	院长名称	是	最长20位
Slogan	口号	否	最长100位

PS：根据字段描述创建数据库表

资源描述（属性操作）

2-1-1、增

操作描述: 添加新的学院信息

操作要素:

要素1: URL = <http://127.0.0.1:8099/api/departments/>

method = POST

要素2: 以 JSON 格式提交学院信息

要素3: 状态码 = 201

响应内容 = 以JSON 格式回显添加的学院信息

操作举例:

访问路径: <http://127.0.0.1:8099/api/departments/> (POST)

提交数据:

```
{  
  "dep_id": "001",  
  "dep_Name": "我的学院",  
  "master_Name": "老张",  
  "slogan": "不睡觉"  
}
```

响应数据:

状态码 201

```
{  
  "dep_id": "001",  
  "dep_Name": "我的学院",  
  "master_Name": "老张",  
  "slogan": "不睡觉"  
}
```

2-1-2、删

操作描述: 根据学院的 id 删除学院信息

操作要素:

要素1: URL = <http://127.0.0.1:8099/api/departments/T01/>

method = DELETE

要素2: 无

要素3: 状态码 = 204

响应内容 = 无

操作举例:

访问路径: <http://127.0.0.1:8099/api/departments/T01/> (DELETE)

提交数据:

无

响应数据:

状态码: 204

内容:无

2-1-3、改

操作描述: 修改学院信息

操作要素:

要素1: URL = <http://127.0.0.1:8099/api/departments/{T01}/>

method = PUT

要素2: 以JSON 格式提交学院信息

要素3:状态码 = 201

响应内容 = 以JSON 格式回显修改的学院信息

操作举例:

访问路径: <http://127.0.0.1:8099/api/departments/T01/> (PUT)

提交数据:

```
{  
  "dep_id": "T01",  
  "dep_Name": "新学院名称",  
  "master_Name": "新院长名称",  
  "slogan": "新口号"  
}
```

响应数据:

```
{  
  "dep_id": "T01",  
  "dep_Name": "新学院名称",  
  "master_Name": "新院长名称",  
  "slogan": "新口号"  
}
```

2-1-4、查

查询所有

操作描述: 查询所有学院信息

操作要素:

要素1: URL = <http://127.0.0.1:8099/api/departments/>

method = GET

要素2: 无

要素3:状态码 = 200

响应内容 = 多条学院信息

操作举例:

访问路径: <http://127.0.0.1:8099/api/departments/> (GET)

提交数据:

无

响应数据:

状态码:200

内容:

```
[  
{  
  "dep_id":"T01",  
  "dep_Name":"新学院名称",  
  "master_Name":"新院长名称",  
  "slogan":"新口号"  
},  
{  
  "dep_id":"T02",  
  "dep_Name":"新学院名称",  
  "master_Name":"新院长名称",  
  "slogan":"新口号"  
},  
....  
]
```

查询指定

操作要素:

要素1: URL = <http://127.0.0.1:8099/api/departments/>{要查询的学院的id}/

method = GET

要素2: 无

要素3:状态码 = 200

响应内容 = 对应的学院信息

操作举例:

访问路径: <http://127.0.0.1:8099/api/departments/T01/> (GET)

提交数据:

无

响应数据:

状态码:200

内容:

```
{  
  "dep_id": "T01",  
  "dep_Name": "新学院名称",  
  "master_Name": "新院长名称",  
  "slogan": "新口号"  
}
```

根据id列表查询

操作要素:

要素1: URL = <http://127.0.0.1:8099/api/departments/>

method = GET

要素2: \$dep_id_list=T01,T02,T03

\$dep_id_list 为键名称, T01,T02,T03 为学院id

要素3:状态码 = 200

响应内容 = 和 id 一一对应的学院具体信息

操作举例:

访问路径: [http://127.0.0.1:8099/api/departments/\\$dep_id_list=T01,T02,T03](http://127.0.0.1:8099/api/departments/$dep_id_list=T01,T02,T03)(GET)

响应数据:

状态码:200

内容:

```
[  
  {  
    "dep_id": "T01",  
    "dep_Name": "学院名称1",  
    "master_Name": "院长名称1",  
    "slogan": "口号1"
```

```
  },
```

```
{  
  "dep_id": "T02",  
  "dep_Name": "学院名称2",  
  "master_Name": "院长名称2",  
  "slogan": "口号2"  
},  
{  
  "dep_id": "T03",  
  "dep_Name": "学院名称3",  
  "master_Name": "院长名称3",  
  "slogan": "口号3"  
}  
]
```

根据院长列表查询

操作要素:

要素1: URL = <http://127.0.0.1:8099/api/departments/>

method = GET

要素2: \$master_name_list=Java-Master,Test-Master

\$master_name_list 为键名称, Java-Master,Test-Master 为院长名称

要素3: 状态码 = 200

响应内容 = 和院长名称 一一对应的学院具体信息

操作举例:

访问路径: <http://127.0.0.1:8099/api/departments/>\$master_name_list=Java-Master,Test-Master(GET)

响应数据:

状态码: 200

内容:

```
[  
{  
  "dep_id": "T01",  
  "dep_Name": "学院名称1",  
  "master_Name": "Java-Master",  
  "slogan": "口号1"
```

```
},  
  
{  
  
"dep_id":"T02",  
  
"dep_Name":"学院名称2",  
  
"master_Name":"Test-Master",  
  
"slogan":"口号2"  
  
}  
  
]
```

模糊查询

操作要素:

要素1: URL = <http://127.0.0.1:8099/api/departments/>

method = GET

要素2: blur=1&dep_name=C

blur=1标志开启模糊查询,非1则不适用模糊查询,dep_name 指根据学院名称执行模糊查询操作

要素3:状态码 = 200

响应内容 = 学院名称包含指定字符的学院的相关信息

操作举例:

访问路径: http://127.0.0.1:8099/api/departments/blur=1&dep_name=C(GET)

响应数据:

状态码:200

内容:

```
[  
  
{  
  
"dep_id":"T100",  
  
"dep_Name":"C语言",  
  
"master_Name":"院长名称100",  
  
"slogan":"口号100"  
  
},  
  
{  
  
"dep_id":"T101",  
  
"dep_Name":"C++",
```

```
"master_Name":"院长名称1001",  
"slogan":"口号101"  
}  
]
```

多条件查询

操作要素:

要素1: URL = <http://127.0.0.1:8099/api/departments/>

method = GET

要素2: slogan=Here is Slogan&master_name=Test-Master&dep_name=Test学院

根据口号, 院长名称学院名称等多条件组合查询学院信息

要素3:状态码 = 200

响应内容 = 条件都复合的学院信息

操作举例:

访问路径: [http://127.0.0.1:8099/api/departments/slogan=Here is Slogan&master_name=Test-Master&dep_name=Test学院\(GET\)](http://127.0.0.1:8099/api/departments/slogan=Here is Slogan&master_name=Test-Master&dep_name=Test学院(GET))

响应数据:

状态码:200

内容:

```
{  
"dep_id":"Test",  
"dep_Name":"Test学院",  
"master_Name":"Test-Master",  
"slogan":"Here is Slogan"  
}
```

补充说明:

三要素释义:

要素1,资源路径以及请求方式

要素2,提交的数据

要素3,响应内容, 只要包含状态码与响应体

特殊符号释义:{} 括起来的数据意味着是可以自定义的, 可变的

提交数据有误时，返回相关错误提示，格式如下：

```
{  
  "status": "状态码",  
  "message": "错误描述"  
}  
####
```