СиММА®

Система многослойного моделирования архитектуры

Описание API

Версия 2.4

ООО «Марк Аврелий»
Marcus Aurelius ltd

Аннотация

Настоящий документ является описанием Application Programming Interface (API) Системы многослойного моделирования архитектуры (СиММА).

Содержание

1 Общие сведения

1.1 Назначение системы

Система моделирования архитектуры предприятия (далее – Система) предназначена для построения описаний (моделей) любых элементов предприятия активной, пассивной и поведенческой природы. Однако не существует никаких ограничений для моделирования любых объектов реальности, репрезентация которых может быть выражена взаимосвязанными каталогами (реестрами) или схемами в строгих и не строгих графических нотациях.

2 Аутентификация для доступа к методам API СиММА

Аутентификация пользователей при доступе к методам API СиММА осуществляется с помощью маркера безопасности, передаваемого в HTTP-запросе в параметре sid.

Чтобы получить значение токена, надо выполнить следующее: ….

3 Описание методов API

3.1 Перечень методов

Компонент СиММА	**Наимено-вание**	**Краткое описание**	**Тип запроса**
Модель	gm	Получить список моделей	GET
Модель	gmi	Получить информацию о статистике модели по ее id	GET
Модель	dmdl	Удалить модель	POST
Класс	gc	Получить список классов определенной модели	GET
Класс	cc	Создать новый класс в определенной модели	POST
Класс	gca	Получить список мета-атрибутов определенного класса	GET
Класс	sc	Задать привязку мета-атрибутов к классу	POST
Мета-атрибут	cattr	Создать новый мета-атрибут	POST
Мета-атрибут	ga	Получить характеристики мета-атрибута	POST
Мета-атрибут	da	Удалить существующий мета-атрибут	POST
Мета-связь	ccnc	Создание новой мета-связи	POST
Мета-связь	gcnc	Просмотр информации о мета-связи по id мета-связи	POST
Элемент	ce	Создать новый элемент в классе	POST
Элемент	ee	Редактировать существующий элемент	POST
Элемент	de	Удалить существующий элемент	POST
Элемент	sevt	Настройка порядка возврата элементов для метода gtvt	POST
Элемент	gtvt	Получить элементы класса	POST
Элемент	gea	Полный просмотр данных по элементу	GET
Связь	cce	Создание связи между элементами классов по ИД элемента	POST
Графин	cdq	Создание графинов на схемах	POST
Схема	cdiag	Создание новой схемы	POST
Стенсил	gs	Получение списка стенсилов	GET

3.2 Примеры описания методов

3.2.1 Создать новый класс.

Header	{"Content-Type": "application/json; charset=utf8"}
Тип запроса	POST
URL запроса	https://customersite.ru:9999/api_json
Path Variable	
Отсутствует	
Request Params	Обязательный
action=cc mid=<id модели> name: 'Название', descr: 'Описание', visible: false/true	Да Да Да Нет Нет
Request Body (json)	
Отсутствует	
Response Param	
Отсутствует	
Response Body (json)	
{"state":o"id созданного класса"} код ответа =0 при успешном завершении	

Пример запроса (POST-запрос)

```
payload = {'action': 'cc', "mid": 'm3', 'name': 'Пробный класс', 'descr':'Пробный класс для документации', 'visible':'True'}

payload = json.dumps(payload)

response = requests.post("http://customersite.ru:9999/api_json", data=payload,

                headers={"Content-Type": "application/json; charset=utf8",

                    'Cookie': 'sid=073f979876664b048878bdc167c2ffaa'})
```

Пример ответа:

```
{'state': '0:o83f'}
```

3.2.2 Создать новый мета-атрибут

Header	{"Content-Type": "application/json; charset=utf8", 'Cookie':'sid= <application sid >' }
Тип запроса	POST
URL запроса	http://customersite.ru:9999/api_json
Path Variable	
Отсутствует	
Request Params	**Обязательный**
action=cattr inputAttrName=<Имя мета-атрибута> inputAttrType:"Тип мета-атрибута" inputDefVal:"значение" inputDefValTA: "значение" inputDefValArr:[] inputAttrId:null inputAttrDescr: "Описание мета-атрибута"	Да Да Да Да Да Да Да
Request Body (json)	
Отсутствует	
Response Param	
Отсутствует	
Response Body (json)	
{state: 0:a<id созданного мета-атрибута>}	

- Возможные ошибки:

- попытка создать атрибут с именем, которое уже существует (возвращает сообщение о дублировании ключа)
- попытка создать элемент, не указав его тип (вернется корректный id созданного атрибута но он не будет доступен).

Пример запроса (POST-запрос)

```
payload = {"action": "cattr",
"inputAttrName":"MyMark","inputAttrType":"s","inputDefVal":"","inputDefValTA":"","inputDefValArr":[],"inputAttrId":"null","inputAttrDescr":""}
payload = json.dumps(payload)
response = requests.post("https://customersite.ru:9999/api_json?mid=m4e",data=payload,
                headers={"Content-Type": "application/json; charset=utf8",
                        'Cookie': 'sid=0e123090417047289a97d0b22b9b83fb'}
                )
```

Пример ответа:

```
{"state":"0:a924"}
```

3.2.3 Создание нового элемента в классе

Позволяет создать новый элемент в классе.

Атрибуты с зарезервированными именами:
«id»- идентификатор элемента,
«name» - наименование элемента,
«descr» - описание элемента.
Внимание: id атрибута - Case-sensitive!

Header	{"Content-Type": "application/json; charset=utf8", 'Cookie':'sid= <application sid >' }
Тип запроса	POST
URL запроса	http://customersite.ru:9999/api_json
Path Variable	
Отсутствует	

Request Params	**Обязательный**
action=ce id=<id класса> data1 = [[name, значение],[id мета-атрибута1, значение],[id мета-атрибута2, значение]]	Да Да Да
Request Body (json)	
Отсутствует	
Response Param	
Отсутствует	
Response Body (json)	
{state: 0:e<id созданного элемента>}	

Пример запроса (POST-запрос)

```
payload = {"action": "ce", "id": “ob3”,
        "data1": [[“name”, “Новый элемент”],
                [“descr”, “Описание нового элемента”],
                [“a674”, “Значение нового элемента”]}
        response = requests.post("http://customersite.ru:9999/api_json?", data=payload,
                headers={"Content-Type": "application/json; charset=utf8",
                        'Cookie':'sid=967476c51c49458bb340b6340c04b1c9'}
                )
```

Пример ответа:

{“state”: “0:e1cb4cc”}

Пример ответа с ошибкой:

{'state': '400 Bad Request: The browser (or proxy) sent a request that this server could not understand.'}

3.2.4 Получить полную информацию по элементу по его id

Возвращает развернутую информацию по элементу. Основной метод для получения информации по модели. Ответ представляет из себя список, состоящий из списков, по количеству равных количеству атрибутов у элемента + 4 обязательных списка. Каждый из списков содержит 12 элементов и принимает значение исходя из того, что он описывает.

Стандартные значения:

Список 1 – содержит 12 атрибутов со сведениями о классе, в рамках которого создан рассматриваемый элемент.

Список 2 – содержит сведения об атрибуте «Наименование» элемента.

Список 3 – содержит сведения об атрибуте «Описание» элемента.

Список 4 – содержит сведения об атрибуте «Присутствие на» элемента (наличие элемента на схемах модели).

Список 5 и далее содержит сведения, описывающие каждый атрибут элемента с уникальным названием.

Header	{"Content-Type": "application/json; charset=utf8", 'Cookie':'sid= <application sid >' }
Тип запроса	GET
URL запроса	http://customersite.ru:9999/api
Path Variable	
Отсутствует	
Request Params	**Обязательный**
action=gea id=<id элемента>	Да Да
Request Body (json)	
Отсутствует	
Response Param	
Отсутствует	
Response Body (json)	
[[],[]…[]] – список списков, с расшифровкой выше.	

Пример запроса (GET-запрос)

```
response = requests.get(“http://customersite.ru:9999/ api?mid=m24&action=gea&id= e1cb4cc",
                headers={"Content-Type": "application/json; charset=utf8",
                        'Cookie': 'sid='967476c51c49458bb340b6340c04b1c9’}
                )
```

Пример ответа:

```
[['Класс', 'Птица', 's', 'obj_name', 'o56', -2, None, 1, [], [], 'e2dea', None, []],
 ['Название', 'Ворона', 's', 'name', 'o56', -1, None, 1, [], [], 'e2dea', None, []],
```

['Описание', '', 't', 'descr', 'o56', 0, None, 1, [], [], 'e2dea', None, []],

['Присутствует на', '', 'r', 'dgi', None, 0, 'Схемы', 3, [], [], 'e2dea', None, []],

['Продолжительность жизни', '', 'i', 'a96', 'o56', 1, None, 1, [], [], 'e2dea', None, []],

['Фото', ' <img src="https://o-prirode.ru/wp-content/webp-express/webp-images/doc-root/wp-content/uploads/2019/08/vorona-3.jpg.webp" alt="Птица ворона: описание, виды, фото, преметы, интересные факты" width="600" height="400"> ', 't', 'a97', 'o56', 2, None, 1, [], [], 'e2dea', None, []],

['Регионы зимовки', '', 'r', 'a10d', 'o56', 3, None, 1, [], [], 'e2dea', {'conn_type': 'p', 'direction': 's', 'id': 'rm4c', 'kind': 'a', 'linked_class': 'o57', 'maxlimit': None, 'name': 'Регионы зимовки', 'opp_name': 'Кто здесь зимует', 'props': {'color': '#FF3300', 'decoration_dst': 'badraw.decoration.connection.DiamondDecorator', 'decoration_src': None}}, []],

['Габариты (длина, высота, ширина), см', 'Средняя полоса', 'e', 'a98', 'o56', 4, None, 1, ['Средняя полоса', 'Юг', 'Север'], [], 'e2dea', None, []],

['Дата прилета, приплода, отлета', '', 'ae', 'a219', 'o56', 5, None, 1, ['На голове', 'на спинке'], [], 'e2dea', None, ['На голове', 'На грудке', 'на спинке']],

‘['Регионы зимовки', '', 'ai', 'a9a', 'o56', 6, None, 1, None, [], 'e2dea', None, []], ‘

['Летние регионы', '', 'ad', 'a9b', 'o56', 7, None, 1, None, [], 'e2dea', None, []],

['Дата рождения', '', 'd', 'a213', 'o56', 8, None, 1, [], [], 'e2dea', None, []],

['Летние регионы', '', 'r', 'a110', 'o56', 9, None, 1, [], [], 'e2dea', {'conn_type': 'p', 'direction': 'd', 'id': 'rm4d', 'kind': 'a', 'linked_class': 'o57', 'maxlimit': None, 'name': 'Летние регионы', 'opp_name': 'Кто здесь выводит птенцов', 'props': {'color': '#FF9900', 'decoration_dst': None, 'decoration_src': 'badraw.decoration.connection.DiamondDecorator'}}, []],

['Отряд', 'Пернатые', 'e', 'a218', 'o56', 10, None, 1, ['Пернатые', 'Хвостатые'], [], 'e2dea', None, []], ['<Автор>', None, 's', 'syscr', None, 10000, 'Системные', 2, [], [], 'e2dea', None, []],

['<Дата создания>', None, 's', 'syscd', None, 10000, 'Системные', 2, [], [], 'e2dea', None, []],

['<Внёс изменения>', None, 's', 'syslu', None, 10000, 'Системные', 2, [], [], 'e2dea', None, []],

['<Дата изменения>', None, 's', 'sysld', None, 10000, 'Системные', 2, [], [], 'e2dea', None, []], ['ID', 'e2dea', 's', 'sysid', None, 10000, 'Системные', 2, [], [], 'e2dea', None, []]]

3.2.5 Редактирование элемента

Позволяет редактировать существующий элемент в классе.

Атрибуты с зарезервированными именами:

«id» - идентификатор элемента,

«name» - наименование элемента,

«descr» - описание элемента.

Внимание: id атрибута - Case-sensitive!

Header	{"Content-Type": "application/json; charset=utf8", 'Cookie':'sid= <application sid >' }
Тип запроса	POST
URL запроса	http://customersite.ru:9999/api_json
Path Variable	
Отсутствует	

Request Params	**Обязательный**
action=ee id=<id элемента> class_id": "id класса" data1 = [[name, значение],[id мета-атрибута1, значение],[id мета-атрибута2, значение]]	Да Да Да Да
Request Body (json)	
Отсутствует	
Response Param	
Отсутствует	
Response Body (json)	
{state: 0:e<id отредактированного элемента>}	

Пример запроса (POST-запрос)

```
payload = {"action":"ee","id":"e8da9c","class_id":"ob2","data1":[
["name","Технические счета. Значение заполнения субконто2"],
["descr","<p>ЗначениеЗаполненияСубконто2</p>"],
["a674","Ссылка"],
["a683","Реквизит"],
["a675","35ef1bd2-25aa-47fc-a735-111d4ce0711c"],
["a682","True"],
["a678","да"]]}
response = requests.post("http://customersite.ru:9999/api_json?", data=payload,
                         headers={"Content-Type": "application/json; charset=utf8",
                               'Cookie':'sid=967476c51c49458bb340b6340c04b1c9'}
                         )
```

Пример ответа:

{'state': '0:e8da9c'}

Пример ответа с ошибкой:

{'state': '400 Bad Request: The browser (or proxy) sent a request that this server could not understand.'}