Плагин для работы с CustomStorage

Плагин CustomStorage

Плагин **CustomStorage** предназначен для работы с кастомными таблицами атрибутов в JavaScript скриптах. Плагин позволяет сохранять и получать данные из кастомных таблиц, аналогично работе с атрибутами лидов через `lead.setAttr()` и `lead.getAttr()` и т.д.

Плагин поддерживает:

- Работу с обычными атрибутами (строки, числа, булевы значения)
- Работу с JSON атрибутами (объекты, массивы, вложенные структуры)
- Точечную нотацию для вложенных JSON структур (например, `user.profile.name`)
- Подкатегории для группировки данных через `key2` и `key3`
- Прямую работу с базой данных без кэширования в памяти

Первоначальная настройка (Setup)

Перед использованием плагина необходимо создать кастомную таблицу для хранения атрибутов. Это можно сделать двумя способами:

Способ 1: Автоматическое создание через метод `setup()`

Метод `setup()` автоматически создает кастомную таблицу со всеми необходимыми полями и уникальным индексом.

Важно: Метод `setup()` можно вызвать только один раз для каждой таблицы. При повторном вызове будет выброшена ошибка, если таблица уже существует.

Пример использования:

```
let customStorage = require('Common.Platform.CustomStorage')

// Создаем новую таблицу для атрибутов
customStorage.setup('my_attributes_table')

// Теперь можно работать с таблицей
customStorage.setAttr('server_name', 'production-server-01')
```

Что делает метод `setup()`:

1. Проверяет, что таблицы с таким именем еще не существует
2. Создает кастомную таблицу с необходимыми полями:
   - `id` - автоинкрементный идентификатор
   - `key` - первый ключ (обязательный)
   - `key2` - второй ключ (опциональный, для подкатегорий)
   - `key3` - третий ключ (опциональный, для подкатегорий)
   - `value_type` - тип значения (`string` или `json`)
   - `value` - значение атрибута
   - `created_at` - дата создания
   - `updated_at` - дата обновления
3. Создает уникальный индекс на комбинацию `(key, key2, key3)` для предотвращения дубликатов

Способ 2: Ручное создание через интерфейс админки

Вы можете создать таблицу вручную через интерфейс админки в разделе **Таблицы**, используя структуру из JSON примера ниже.

Структура таблицы (JSON для импорта):

```
{
    "version": "2.50.0",
    "format": 2,
    "custom_tables": {
        "business_attributes": {
```

```
            "is_enabled": 1,
            "name": "business_attributes",
            "title": "Атрибуты бизнеса",
            "comment": null,
            "is_sync_struct": 1,
            "is_show_in_menu": 1,
            "has_api_access": 0,
            "sort_order": 0,
            "is_compact_view": 0,
            "line_height": 1,
            "max_width_px": null,
            "css": null,
            "js": null,
            "sort_order_data": 1,
            "fields": {
                "id": {
                    "is_enabled": 1,
                    "name": "id",
                    "title": "ID",
                    "type": "AUTOINC",
                    "is_required": 1,
                    "size": null,
                    "default_value": null,
                    "hint": null,
                    "tooltip": null,
                    "sort_order": 0,
                    "precision": null,
                    "is_sync_struct": 1,
                    "is_locked_for_regular_user": 0,
                    "max_width_px": null
                },
                "key": {
                    "is_enabled": 1,
                    "name": "key",
                    "title": "Первый ключ",
                    "type": "TEXT",
                    "is_required": 1,
                    "size": null,
                    "default_value": "",
                    "hint": null,
```

```
            "tooltip": null,
            "sort_order": 2,
            "precision": null,
            "is_sync_struct": 1,
            "is_locked_for_regular_user": 0,
            "max_width_px": null
        },
        "key2": {
            "is_enabled": 1,
            "name": "key2",
            "title": "Второй ключ",
            "type": "TEXT",
            "is_required": 0,
            "size": null,
            "default_value": "",
            "hint": null,
            "tooltip": null,
            "sort_order": 3,
            "precision": null,
            "is_sync_struct": 1,
            "is_locked_for_regular_user": 0,
            "max_width_px": null
        },
        "key3": {
            "is_enabled": 1,
            "name": "key3",
            "title": "Третий ключ",
            "type": "TEXT",
            "is_required": 0,
            "size": null,
            "default_value": "",
            "hint": null,
            "tooltip": null,
            "sort_order": 4,
            "precision": null,
            "is_sync_struct": 1,
            "is_locked_for_regular_user": 0,
            "max_width_px": null
        },
        "value_type": {
```

```
            "is_enabled": 1,
            "name": "value_type",
            "title": "Тип значения",
            "type": "TEXT",
            "is_required": 0,
            "size": null,
            "default_value": "",
            "hint": null,
            "tooltip": null,
            "sort_order": 5,
            "precision": null,
            "is_sync_struct": 1,
            "is_locked_for_regular_user": 0,
            "max_width_px": null
        },
        "value": {
            "is_enabled": 1,
            "name": "value",
            "title": "Значение",
            "type": "TEXT",
            "is_required": 1,
            "size": null,
            "default_value": "",
            "hint": null,
            "tooltip": null,
            "sort_order": 6,
            "precision": null,
            "is_sync_struct": 1,
            "is_locked_for_regular_user": 0,
            "max_width_px": null
        },
        "created_at": {
            "is_enabled": 1,
            "name": "created_at",
            "title": "Дата создания",
            "type": "CREATED_AT",
            "is_required": 1,
            "size": null,
            "default_value": "NOW",
            "hint": null,
```

```
                    "tooltip": null,
                    "sort_order": 7,
                    "precision": null,
                    "is_sync_struct": 1,
                    "is_locked_for_regular_user": 0,
                    "max_width_px": null
                },
                "updated_at": {
                    "is_enabled": 1,
                    "name": "updated_at",
                    "title": "Дата изменения",
                    "type": "UPDATED_AT",
                    "is_required": 0,
                    "size": null,
                    "default_value": null,
                    "hint": null,
                    "tooltip": null,
                    "sort_order": 8,
                    "precision": null,
                    "is_sync_struct": 1,
                    "is_locked_for_regular_user": 0,
                    "max_width_px": null
                }
            }
        }
    }
}
```


Важно: После создания таблицы вручную необходимо создать уникальный индекс на комбинацию `(key, key2, key3)` в базе данных.

Подключение плагина

Для использования плагина в JavaScript скрипте необходимо подключить его через `require()`:

```
let customStorage = require('Common.Platform.CustomStorage')
```

После подключения необходимо установить имя таблицы, с которой будет работать плагин:

```
customStorage.setTableName('business_attributes')
```

Методы работы с таблицей

setTableName()

Устанавливает имя кастомной таблицы для работы.

Сигнатура:

```
customStorage.setTableName(string $tableName): self
```

Параметры:

- `$tableName` (string) - Имя кастомной таблицы, которая должна существовать в текущем бизнесе

Возвращает: `self` - Возвращает сам объект для цепочки вызовов

Пример:

```
customStorage.setTableName('business_attributes')
```

---

getTableName()

Возвращает имя текущей установленной таблицы.

Сигнатура:

```
customStorage.getTableName(): string|null
```

Возвращает: `string|null` - Имя таблицы или `null`, если таблица не установлена

Пример:

```
let tableName = customStorage.getTableName()
bot.sendText('Работаем с таблицей: ' + tableName)
```

---

setup()

Создает новую кастомную таблицу для атрибутов со всеми необходимыми полями и уникальным индексом.

Сигнатура:

```
customStorage.setup(string $tableName): self
```

Параметры:

- `$tableName` (string) - Имя создаваемой таблицы

Возвращает: `self` - Возвращает сам объект для цепочки вызовов

Исключения:

- Выбрасывает ошибку, если таблица с таким именем уже существует
- Выбрасывает ошибку, если не указан Business ID

Пример:

```
// Создаем новую таблицу
customStorage.setup('my_attributes_table')

// Теперь можно работать с ней
customStorage.setAttr('test_key', 'test_value')
```

Важно: Метод `setup()` автоматически устанавливает созданную таблицу как текущую, поэтому после вызова `setup()` можно сразу начинать работу с атрибутами.

Методы работы с обычными атрибутами

setAttr()

Устанавливает значение обычного атрибута в базе данных.

Сигнатура:

```
customStorage.setAttr(string $key, mixed $value, ?string $key2 = null, ?string $key3 = null): self
```

Параметры:

- `$key` (string) - Ключ атрибута (обязательный)
- `$value` (mixed) - Значение атрибута (строка, число, булево значение)
- `$key2` (string|null) - Второй ключ для подкатегории (опциональный)
- `$key3` (string|null) - Третий ключ для подкатегории (опциональный)

Возвращает: `self` - Возвращает сам объект для цепочки вызовов

Примеры:

```
// Простое сохранение
customStorage.setAttr('server_name', 'production-server-01')
customStorage.setAttr('server_port', '8080')

// С подкатегориями
customStorage.setAttr('server_name', 'server-01', 'region1', 'dc1')
customStorage.setAttr('server_ip', '192.168.1.1', 'region1', 'dc1')
```

Примечание: Если для комбинации `(key, key2, key3)` уже существует запись, она будет обновлена. Если найдено несколько записей, будет выброшена ошибка.

---

getAttr()

Получает значение обычного атрибута из базы данных.

Сигнатура:

```
customStorage.getAttr(string $key, ?string $key2 = null, ?string $key3 = null): mixed|null
```

Параметры:

- `$key` (string) - Ключ атрибута (обязательный)
- `$key2` (string|null) - Второй ключ для подкатегории (опциональный)
- `$key3` (string|null) - Третий ключ для подкатегории (опциональный)

Возвращает: `mixed|null` - Значение атрибута или `null`, если атрибут не найден

Примеры:

```
// Простое получение
let serverName = customStorage.getAttr('server_name')

// С подкатегориями
let serverName = customStorage.getAttr('server_name', 'region1', 'dc1')
let serverIp = customStorage.getAttr('server_ip', 'region1', 'dc1')
```

Примечание: Если для комбинации `(key, key2, key3)` найдено несколько записей, будет выброшена ошибка.

---

getIntAttr()

Получает значение атрибута, преобразованное в целое число.

Сигнатура:

```
customStorage.getIntAttr(string $key, ?int $default = 0, ?string $key2 = null, ?string $key3 = null): int|null
```

Параметры:

- `$key` (string) - Ключ атрибута (обязательный)
- `$default` (int|null) - Значение по умолчанию, если атрибут не найден (по умолчанию: `0`)
- `$key2` (string|null) - Второй ключ для подкатегории (опциональный)
- `$key3` (string|null) - Третий ключ для подкатегории (опциональный)

Возвращает: `int|null` - Целое число или значение по умолчанию

Примеры:

```
// Получение с значением по умолчанию
let port = customStorage.getIntAttr('server_port', 8080)
let nonExistent = customStorage.getIntAttr('non_existent', 100) // вернет 100
```

```
// С подкатегориями
let port = customStorage.getIntAttr('server_port', 8080, 'region1', 'dc1')
```

---

getFloatAttr()

Получает значение атрибута, преобразованное в число с плавающей точкой.

Сигнатура:

```
customStorage.getFloatAttr(string $key, ?float $default = 0.0, ?string $key2 = null, ?string $key3 = null): float|null
```

Параметры:

- `$key` (string) - Ключ атрибута (обязательный)
- `$default` (float|null) - Значение по умолчанию, если атрибут не найден (по умолчанию: `0.0`)
- `$key2` (string|null) - Второй ключ для подкатегории (опциональный)
- `$key3` (string|null) - Третий ключ для подкатегории (опциональный)

Возвращает: `float|null` - Число с плавающей точкой или значение по умолчанию

Примеры:

```
let price = customStorage.getFloatAttr('price', 0.0)
let nonExistent = customStorage.getFloatAttr('non_existent', 2.5) // вернет 2.5
```

---

getBoolAttr()

Получает значение атрибута, преобразованное в булево значение.

Сигнатура:

```
customStorage.getBoolAttr(string $key, ?bool $default = false, ?string $key2 = null, ?string $key3 = null): bool|null
```

Параметры:

- `$key` (string) - Ключ атрибута (обязательный)

- `$default` (bool|null) - Значение по умолчанию, если атрибут не найден (по умолчанию: `false`)
- `$key2` (string|null) - Второй ключ для подкатегории (опциональный)
- `$key3` (string|null) - Третий ключ для подкатегории (опциональный)

Возвращает: `bool|null` - Булево значение или значение по умолчанию

Особенности обработки строковых значений:

- Строки `'false'`, `'0'`, `''`, `'no'`, `'off'` преобразуются в `false`
- Строки `'true'`, `'1'`, `'yes'`, `'on'` преобразуются в `true`
- Остальные строки преобразуются по стандартным правилам PHP

Примеры:

```
let isActive = customStorage.getBoolAttr('is_active', false)
let nonExistent = customStorage.getBoolAttr('non_existent', true) // вернет true

// Строковые значения обрабатываются корректно
customStorage.setAttr('bool_true', 'true')
customStorage.setAttr('bool_false', 'false')
let val1 = customStorage.getBoolAttr('bool_true') // вернет true
let val2 = customStorage.getBoolAttr('bool_false') // вернет false
```

---

isAttrExist()

Проверяет существование атрибута в базе данных.

Сигнатура:

```
customStorage.isAttrExist(string $key, ?string $key2 = null, ?string $key3 = null): bool
```

Параметры:

- `$key` (string) - Ключ атрибута (обязательный)
- `$key2` (string|null) - Второй ключ для подкатегории (опциональный)
- `$key3` (string|null) - Третий ключ для подкатегории (опциональный)

Возвращает: `bool` - `true`, если атрибут существует, `false` - если не существует

Примеры:

```
if (customStorage.isAttrExist('server_name')) {
    bot.sendText('Сервер найден: ' + customStorage.getAttr('server_name'))
} else {
    bot.sendText('Сервер не найден')
}

// С подкатегориями
if (customStorage.isAttrExist('server_name', 'region1', 'dc1')) {
    bot.sendText('Сервер найден в регионе 1')
}
```

---

issetAttr()

Проверяет существование атрибута в базе данных (аналог `isAttrExist()`).

Сигнатура:

```
customStorage.issetAttr(string $key, ?string $key2 = null, ?string $key3 = null): bool
```

Параметры:

- `$key` (string) - Ключ атрибута (обязательный)
- `$key2` (string|null) - Второй ключ для подкатегории (опциональный)
- `$key3` (string|null) - Третий ключ для подкатегории (опциональный)

Возвращает: `bool` - `true`, если атрибут существует, `false` - если не существует

Примеры:

```
if (customStorage.issetAttr('server_name')) {
    bot.sendText('Сервер найден')
}
```

---

deleteAttr()

Удаляет атрибут из базы данных.

Сигнатура:

```
customStorage.deleteAttr(string $key, ?string $key2 = null, ?string $key3 = null): self
```

Параметры:

- `$key` (string) - Ключ атрибута (обязательный)
- `$key2` (string|null) - Второй ключ для подкатегории (опциональный)
- `$key3` (string|null) - Третий ключ для подкатегории (опциональный)

Возвращает: `self` - Возвращает сам объект для цепочки вызовов

Примеры:

```
// Удаление простого атрибута
customStorage.deleteAttr('server_port')

// Удаление с подкатегориями
customStorage.deleteAttr('server_name', 'region1', 'dc1')
```

---

getAllAttr()

Получает все обычные атрибуты (не JSON) из базы данных.

Сигнатура:

```
customStorage.getAllAttr(): array
```

Возвращает: `array` - Ассоциативный массив всех атрибутов, где ключ - это комбинация `key`, `key2`, `key3` (разделенные точками), а значение - значение атрибута

Примеры:

```
let allAttrs = customStorage.getAllAttr()
bot.sendText('Всего атрибутов: ' + Object.keys(allAttrs).length)

// Пример структуры результата:
// {
//     "server_name": "server-01",
//     "server_name.region1.dc1": "server-01",
//     "server_ip.region1.dc1": "192.168.1.1"
// }
```

Методы работы с JSON атрибутами

setJsonAttr()

Устанавливает значение JSON атрибута в базе данных. Поддерживает точечную нотацию для вложенных структур.

Сигнатура:

```
customStorage.setJsonAttr(string $key, mixed $value, ?string $key2 = null, ?string $key3 = null): self
```

Параметры:

- `$key` (string) - Ключ атрибута. Может использоваться точечная нотация для вложенных структур (например, `user.profile.name`)
- `$value` (mixed) - Значение для установки. Может быть объектом, массивом, строкой, числом, булевым значением
- `$key2` (string|null) - Второй ключ для подкатегории (опциональный, не связан с вложенностью JSON)
- `$key3` (string|null) - Третий ключ для подкатегории (опциональный, не связан с вложенностью JSON)

Возвращает: `self` - Возвращает сам объект для цепочки вызовов

Особенности:

- **Точечная нотация:** При использовании точечной нотации (например, `user.profile.name`) значение устанавливается внутрь существующего JSON объекта. Верхний ключ (в данном случае `user`) используется как `key` в БД, а остальные части пути создают вложенную структуру.
- **Без точечной нотации:** Если точечная нотация не используется, значение должно быть массивом или объектом (для верхнего уровня).
- **С точечной нотацией:** Разрешены любые типы значений (строки, числа, bool, массивы, объекты).

Примеры:

```
// Простой JSON объект
customStorage.setJsonAttr('config', {
    timeout: 30,
    retries: 3,
    enabled: true
})

// Точечная нотация для вложенных значений
customStorage.setJsonAttr('user.profile.name', 'John Doe')
customStorage.setJsonAttr('user.profile.email', 'john@example.com')
customStorage.setJsonAttr('user.profile.age', 30)
customStorage.setJsonAttr('user.settings.theme', 'dark')

// Глубокая вложенность
customStorage.setJsonAttr('app.settings.database.host', 'localhost')
customStorage.setJsonAttr('app.settings.database.port', 5432)
customStorage.setJsonAttr('app.settings.cache.enabled', true)

// С подкатегориями (key2, key3)
customStorage.setJsonAttr('server.config', { cpu: 8, ram: 32 }, 'region1', 'dc1')
customStorage.setJsonAttr('server.config', { cpu: 16, ram: 64 }, 'region2', 'dc2')

// Массивы
customStorage.setJsonAttr('packages', [
    { name: 'nginx', version: '1.18.0' },
    { name: 'php', version: '8.1.0' },
    { name: 'mysql', version: '8.0.0' }
])
```

Примечание: Если для комбинации `(key, key2, key3)` уже существует запись, JSON объект будет обновлен. Если найдено несколько записей, будет выброшена ошибка.

---

getJsonAttr()

Получает значение JSON атрибута из базы данных. Поддерживает точечную нотацию для извлечения вложенных значений.

Сигнатура:

```
customStorage.getJsonAttr(string $key, ?string $key2 = null, ?string $key3 = null): mixed|null
```

Параметры:

- `$key` (string) - Ключ атрибута. Может использоваться точечная нотация для вложенных структур (например, `user.profile.name`)
- `$key2` (string|null) - Второй ключ для подкатегории (опциональный)
- `$key3` (string|null) - Третий ключ для подкатегории (опциональный)

Возвращает: `mixed|null` - Значение JSON атрибута или `null`, если атрибут не найден

Примеры:

```
// Получение простого JSON объекта
let config = customStorage.getJsonAttr('config')
if (config) {
    bot.sendText('Timeout: ' + config.timeout)
    bot.sendText('Retries: ' + config.retries)
}

// Получение вложенных значений через точечную нотацию
let userName = customStorage.getJsonAttr('user.profile.name')
let userEmail = customStorage.getJsonAttr('user.profile.email')
let userAge = customStorage.getJsonAttr('user.profile.age')

// Получение всего объекта верхнего уровня
let userProfile = customStorage.getJsonAttr('user')
// userProfile будет содержать: { profile: { name: "John Doe", email: "john@example.com", age:
30 }, settings: { theme: "dark" } }

// Глубокая вложенность
let dbHost = customStorage.getJsonAttr('app.settings.database.host')
let dbPort = customStorage.getJsonAttr('app.settings.database.port')

// С подкатегориями
let serverConfig = customStorage.getJsonAttr('server.config', 'region1', 'dc1')
if (serverConfig) {
    bot.sendText('CPU: ' + serverConfig.cpu + ' cores')
    bot.sendText('RAM: ' + serverConfig.ram + ' GB')
}
```

Примечание: Если для комбинации `(key, key2, key3)` найдено несколько записей, будет выброшена ошибка.

---

isJsonAttrKeyExist()

Проверяет существование JSON атрибута в базе данных.

Сигнатура:

```
customStorage.isJsonAttrKeyExist(string $key, ?string $key2 = null, ?string $key3 = null): bool
```

Параметры:

- `$key` (string) - Ключ атрибута. Может использоваться точечная нотация (например, `user.profile.name`)
- `$key2` (string|null) - Второй ключ для подкатегории (опциональный)
- `$key3` (string|null) - Третий ключ для подкатегории (опциональный)

Возвращает: `bool` - `true`, если JSON атрибут существует, `false` - если не существует

Примеры:

```
if (customStorage.isJsonAttrKeyExist('user.profile.name')) {
    bot.sendText('Имя пользователя: ' + customStorage.getJsonAttr('user.profile.name'))
}

// С подкатегориями
if (customStorage.isJsonAttrKeyExist('server.config', 'region1', 'dc1')) {
    bot.sendText('Конфигурация сервера найдена')
}
```

---

issetJsonAttr()

Проверяет существование JSON атрибута в базе данных (аналог `isJsonAttrKeyExist()`).

Сигнатура:

```
customStorage.issetJsonAttr(string $key, ?string $key2 = null, ?string $key3 = null): bool
```

Параметры:

- `$key` (string) - Ключ атрибута. Может использоваться точечная нотация (например, `user.profile.name`)
- `$key2` (string|null) - Второй ключ для подкатегории (опциональный)
- `$key3` (string|null) - Третий ключ для подкатегории (опциональный)

Возвращает: `bool` - `true`, если JSON атрибут существует, `false` - если не существует

Примеры:

```
if (customStorage.issetJsonAttr('user.profile.email')) {
    bot.sendText('Email установлен')
}
```

---

deleteJsonAttr()

Удаляет JSON атрибут из базы данных.

Сигнатура:

```
customStorage.deleteJsonAttr(string $key, ?string $key2 = null, ?string $key3 = null): self
```

Параметры:

- `$key` (string) - Ключ атрибута. Может использоваться точечная нотация (например, `user.profile.name`)
- `$key2` (string|null) - Второй ключ для подкатегории (опциональный)
- `$key3` (string|null) - Третий ключ для подкатегории (опциональный)

Возвращает: `self` - Возвращает сам объект для цепочки вызовов

Важно: При удалении через точечную нотацию удаляется весь объект верхнего уровня. Например, `deleteJsonAttr('user')` удалит всю запись с `key='user'`, включая все вложенные значения (`user.profile.name`, `user.profile.email` и т.д.).

Примеры:

```
// Удаление простого JSON атрибута
customStorage.deleteJsonAttr('config')

// Удаление вложенного JSON атрибута (удаляет весь верхний уровень)
customStorage.deleteJsonAttr('user') // Удалит всю запись с key='user'
```

```
// Удаление с подкатегориями
customStorage.deleteJsonAttr('server.config', 'region1', 'dc1')
```

---

getAllJsonAttrs()

Получает все JSON атрибуты из базы данных.

Сигнатура:

```
customStorage.getAllJsonAttrs(): array
```

Возвращает: `array` - Ассоциативный массив всех JSON атрибутов, где ключ - это комбинация `key`, `key2`, `key3` (разделенные точками), а значение - распарсенный JSON объект

Примеры:

```
let allJsonAttrs = customStorage.getAllJsonAttrs()
bot.sendText('Всего JSON атрибутов: ' + Object.keys(allJsonAttrs).length)

// Пример структуры результата:
// {
//     "config": { timeout: 30, retries: 3, enabled: true },
//     "user": { profile: { name: "John Doe", email: "john@example.com" } },
//     "server.region1.dc1": { config: { cpu: 8, ram: 32 } }
// }
```

Полный справочник методов

Методы работы с таблицей

Метод	Описание
setTableName(string $tableName): self	Устанавливает имя кастомной таблицы для работы
getTableName(): string|null	Возвращает имя текущей установленной таблицы

setup(string $tableName): self	Создает новую кастомную таблицу для атрибутов со всеми необходимыми полями и уникальным индексом

Методы работы с обычными атрибутами

Метод	Описание
setAttr(string $key, mixed $value, ?string $key2 = null, ?string $key3 = null): self	Устанавливает значение обычного атрибута в базе данных
getAttr(string $key, ?string $key2 = null, ?string $key3 = null): mixed|null	Получает значение обычного атрибута из базы данных
getIntAttr(string $key, ?int $default = 0, ?string $key2 = null, ?string $key3 = null): int|null	Получает значение атрибута, преобразованное в целое число
getFloatAttr(string $key, ?float $default = 0.0, ?string $key2 = null, ?string $key3 = null): float|null	Получает значение атрибута, преобразованное в число с плавающей точкой
getBoolAttr(string $key, ?bool $default = false, ?string $key2 = null, ?string $key3 = null): bool|null	Получает значение атрибута, преобразованное в булево значение
isAttrExist(string $key, ?string $key2 = null, ?string $key3 = null): bool	Проверяет существование атрибута в базе данных
issetAttr(string $key, ?string $key2 = null, ?string $key3 = null): bool	Проверяет существование атрибута в базе данных (аналог isAttrExist)
deleteAttr(string $key, ?string $key2 = null, ?string $key3 = null): self	Удаляет атрибут из базы данных
getAllAttr(): array	Получает все обычные атрибуты (не JSON) из базы данных

Методы работы с JSON атрибутами

Метод	Описание
setJsonAttr(string $key, mixed $value, ?string $key2 = null, ?string $key3 = null): self	Устанавливает значение JSON атрибута в базе данных. Поддерживает точечную нотацию для вложенных структур
getJsonAttr(string $key, ?string $key2 = null, ?string $key3 = null): mixed|null	Получает значение JSON атрибута из базы данных. Поддерживает точечную нотацию для извлечения вложенных значений
isJsonAttrKeyExist(string $key, ?string $key2 = null, ?string $key3 = null): bool	Проверяет существование JSON атрибута в базе данных
issetJsonAttr(string $key, ?string $key2 = null, ?string $key3 = null): bool	Проверяет существование JSON атрибута в базе данных (аналог isJsonAttrKeyExist)

deleteJsonAttr(string $key, ?string $key2 = null, ?string $key3 = null): self	Удаляет JSON атрибут из базы данных
getAllJsonAttrs(): array	Получает все JSON атрибуты из базы данных

Примеры использования

Пример 1: Базовое использование

```
let customStorage = require('Common.Platform.CustomStorage')

// Устанавливаем таблицу
customStorage.setTableName('business_attributes')

// Сохранение простых атрибутов
customStorage.setAttr('server_name', 'production-server-01')
customStorage.setAttr('server_ip', '192.168.1.100')
customStorage.setAttr('server_port', '8080')

// Получение атрибутов
let serverName = customStorage.getAttr('server_name')
let serverIp = customStorage.getAttr('server_ip')
let port = customStorage.getIntAttr('server_port', 0)

bot.sendText('Сервер: ' + serverName + ' (' + serverIp + ':' + port + ')')
```

Пример 2: Использование подкатегорий (key2, key3)

```
let customStorage = require('Common.Platform.CustomStorage')
customStorage.setTableName('business_attributes')

// Сохранение данных для разных регионов и датацентров
customStorage.setAttr('server_name', 'server-01', 'us-east', 'dc-01')
customStorage.setAttr('server_ip', '10.0.1.10', 'us-east', 'dc-01')
```

```
customStorage.setAttr('server_name', 'server-02', 'us-east', 'dc-02')
customStorage.setAttr('server_ip', '10.0.2.10', 'us-east', 'dc-02')

// Получение данных для конкретного региона и датацентра
let serverName = customStorage.getAttr('server_name', 'us-east', 'dc-01')
let serverIp = customStorage.getAttr('server_ip', 'us-east', 'dc-01')

bot.sendText('Сервер в us-east/dc-01: ' + serverName + ' (' + serverIp + ')')
```

Пример 3: Работа с JSON атрибутами

```
let customStorage = require('Common.Platform.CustomStorage')
customStorage.setTableName('business_attributes')

// Сохранение простого JSON объекта
customStorage.setJsonAttr('config', {
    timeout: 30,
    retries: 3,
    enabled: true
})

// Получение JSON объекта
let config = customStorage.getJsonAttr('config')
if (config) {
    bot.sendText('Timeout: ' + config.timeout)
    bot.sendText('Retries: ' + config.retries)
    bot.sendText('Enabled: ' + config.enabled)
}
```

Пример 4: Точечная нотация для вложенных JSON структур

```
let customStorage = require('Common.Platform.CustomStorage')
customStorage.setTableName('business_attributes')

// Установка вложенных значений через точечную нотацию
```

```
customStorage.setJsonAttr('user.profile.name', 'John Doe')
customStorage.setJsonAttr('user.profile.email', 'john@example.com')
customStorage.setJsonAttr('user.profile.age', 30)
customStorage.setJsonAttr('user.settings.theme', 'dark')
customStorage.setJsonAttr('user.settings.language', 'ru')

// Получение вложенных значений
let userName = customStorage.getJsonAttr('user.profile.name')
let userEmail = customStorage.getJsonAttr('user.profile.email')
let userTheme = customStorage.getJsonAttr('user.settings.theme')

bot.sendText('Пользователь: ' + userName + ' (' + userEmail + ')')
bot.sendText('Тема: ' + userTheme)

// Получение всего объекта верхнего уровня
let userProfile = customStorage.getJsonAttr('user')
// userProfile будет содержать: { profile: { name: "John Doe", email: "john@example.com", age:
30 }, settings: { theme: "dark", language: "ru" } }
```

Пример 5: JSON атрибуты с подкатегориями

```
let customStorage = require('Common.Platform.CustomStorage')
customStorage.setTableName('business_attributes')

// Сохранение JSON конфигурации для разных регионов
customStorage.setJsonAttr('server.config', { cpu: 8, ram: 32 }, 'region1', 'dc1')
customStorage.setJsonAttr('server.config', { cpu: 16, ram: 64 }, 'region2', 'dc2')

// Получение конфигурации для конкретного региона
let config1 = customStorage.getJsonAttr('server.config', 'region1', 'dc1')
let config2 = customStorage.getJsonAttr('server.config', 'region2', 'dc2')

if (config1) {
    bot.sendText('Регион 1: CPU=' + config1.cpu + ', RAM=' + config1.ram)
}
if (config2) {
```

```
    bot.sendText('Регион 2: CPU=' + config2.cpu + ', RAM=' + config2.ram)
}
```

Пример 6: Глубокая вложенность JSON

```
let customStorage = require('Common.Platform.CustomStorage')
customStorage.setTableName('business_attributes')

// Установка глубоко вложенных значений
customStorage.setJsonAttr('app.settings.database.host', 'localhost')
customStorage.setJsonAttr('app.settings.database.port', 5432)
customStorage.setJsonAttr('app.settings.database.name', 'myapp')
customStorage.setJsonAttr('app.settings.cache.enabled', true)
customStorage.setJsonAttr('app.settings.cache.ttl', 3600)

// Получение глубоко вложенных значений
let dbHost = customStorage.getJsonAttr('app.settings.database.host')
let dbPort = customStorage.getJsonAttr('app.settings.database.port')
let cacheEnabled = customStorage.getJsonAttr('app.settings.cache.enabled')

bot.sendText('DB Host: ' + dbHost)
bot.sendText('DB Port: ' + dbPort)
bot.sendText('Cache Enabled: ' + cacheEnabled)
```

Пример 7: Работа с массивами

```
let customStorage = require('Common.Platform.CustomStorage')
customStorage.setTableName('business_attributes')

// Сохранение массива
customStorage.setJsonAttr('packages', [
    { name: 'nginx', version: '1.18.0' },
    { name: 'php', version: '8.1.0' },
    { name: 'mysql', version: '8.0.0' }
])

// Получение массива
```

```
let packages = customStorage.getJsonAttr('packages')
if (packages && Array.isArray(packages)) {
    bot.sendText('Установлено пакетов: ' + packages.length)
    packages.forEach(function(pkg) {
        bot.sendText('- ' + pkg.name + ' v' + pkg.version)
    })
}
```

Пример 8: Проверка существования и удаление

```
let customStorage = require('Common.Platform.CustomStorage')
customStorage.setTableName('business_attributes')

// Проверка существования
if (customStorage.isAttrExist('server_name')) {
    let serverName = customStorage.getAttr('server_name')
    bot.sendText('Сервер найден: ' + serverName)
} else {
    bot.sendText('Сервер не найден')
}

// Проверка JSON атрибута
if (customStorage.isJsonAttrKeyExist('user.profile.name')) {
    let userName = customStorage.getJsonAttr('user.profile.name')
    bot.sendText('Имя пользователя: ' + userName)
}

// Удаление атрибутов
customStorage.deleteAttr('server_port')
customStorage.deleteJsonAttr('config')

// Удаление с подкатегориями
customStorage.deleteAttr('server_name', 'region1', 'dc1')
customStorage.deleteJsonAttr('server.config', 'region1', 'dc1')
```

Пример 9: Массовые операции

```
let customStorage = require('Common.Platform.CustomStorage')
customStorage.setTableName('business_attributes')

// Получение всех обычных атрибутов
let allAttrs = customStorage.getAllAttr()
bot.sendText('Всего обычных атрибутов: ' + Object.keys(allAttrs).length)

// Получение всех JSON атрибутов
let allJsonAttrs = customStorage.getAllJsonAttrs()
bot.sendText('Всего JSON атрибутов: ' + Object.keys(allJsonAttrs).length)
```

Важные замечания

Безопасность

- Плагин работает только с кастомными таблицами текущего бизнеса
- Все операции проверяют принадлежность таблицы к текущему бизнесу
- При попытке доступа к таблице другого бизнеса будет выброшена ошибка

Уникальность записей

- В таблице создается уникальный индекс на комбинацию `(key, key2, key3)`
- При попытке установить значение для существующей комбинации `(key, key2, key3)` запись будет обновлена
- Если в базе данных найдено несколько записей для одной комбинации `(key, key2, key3)`, будет выброшена ошибка с просьбой вручную разрешить дубликаты

Работа с памятью

- Плагин работает напрямую с базой данных без кэширования в памяти
- Все операции чтения и записи выполняются непосредственно в БД
- Это обеспечивает актуальность данных, но может быть медленнее при частых обращениях

Точечная нотация для JSON

- При использовании точечной нотации (например, `user.profile.name`) верхний ключ (`user`) используется как `key` в БД
- Остальные части пути (`profile`, `name`) создают вложенную структуру в JSON объекте
- При установке нескольких значений с одним верхним ключом они объединяются в один JSON объект

Подкатегории key2 и key3

- `key2` и `key3` используются для группировки данных и не связаны с вложенностью JSON
- Они позволяют хранить разные значения для одного и того же `key` в разных контекстах
- Например, можно хранить конфигурацию сервера для разных регионов: `server.config` с `key2='region1'`, `key3='dc1'`

Обработка ошибок

Все методы плагина могут выбрасывать исключения типа `UnauthotizedV8Exception` в следующих случаях:

- Таблица не найдена
- Таблица не принадлежит текущему бизнесу
- Найдено несколько записей для одной комбинации `(key, key2, key3)`
- Ошибка при работе с базой данных

Важно: В V8Js невозможно отлавливать исключения бэкенда через `try/catch` в JavaScript. Поэтому рекомендуется использовать специальные методы проверки существования атрибутов перед их использованием:

Проверка существования обычных атрибутов

Для проверки существования обычных атрибутов используйте методы `isAttrExist()` или `issetAttr()`:

```
let customStorage = require('Common.Platform.CustomStorage')
customStorage.setTableName('business_attributes')
```

```
// Безопасное получение атрибута с проверкой существования
if (customStorage.isAttrExist('server_name')) {
    let value = customStorage.getAttr('server_name')
    bot.sendText('Значение: ' + value)
} else {
    bot.sendText('Атрибут не найден')
}

// Альтернативный вариант с issetAttr()
if (customStorage.issetAttr('server_name')) {
    let value = customStorage.getAttr('server_name')
    bot.sendText('Значение: ' + value)
}

// С подкатегориями
if (customStorage.isAttrExist('server_name', 'region1', 'dc1')) {
    let value = customStorage.getAttr('server_name', 'region1', 'dc1')
    bot.sendText('Сервер в регионе 1: ' + value)
}
```

Проверка существования JSON атрибутов

Для проверки существования JSON атрибутов используйте методы `isJsonAttrKeyExist()` или `issetJsonAttr()`:

```
let customStorage = require('Common.Platform.CustomStorage')
customStorage.setTableName('business_attributes')

// Проверка простого JSON атрибута
if (customStorage.isJsonAttrKeyExist('config')) {
    let config = customStorage.getJsonAttr('config')
    bot.sendText('Timeout: ' + config.timeout)
}

// Проверка вложенного JSON атрибута с точечной нотацией
if (customStorage.isJsonAttrKeyExist('user.profile.name')) {
    let userName = customStorage.getJsonAttr('user.profile.name')
```

```
    bot.sendText('Имя пользователя: ' + userName)
}

// Альтернативный вариант с issetJsonAttr()
if (customStorage.issetJsonAttr('user.profile.email')) {
    let email = customStorage.getJsonAttr('user.profile.email')
    bot.sendText('Email: ' + email)
}

// С подкатегориями
if (customStorage.isJsonAttrKeyExist('server.config', 'region1', 'dc1')) {
    let config = customStorage.getJsonAttr('server.config', 'region1', 'dc1')
    bot.sendText('CPU: ' + config.cpu + ' cores')
}
```

Рекомендация: Всегда используйте методы проверки существования (`isAttrExist()`, `issetAttr()`, `isJsonAttrKeyExist()`, `issetJsonAttr()`) вместо проверки на `null` после вызова `getAttr()` или `getJsonAttr()`. Это более явный и безопасный способ проверки.

Сравнение с атрибутами лидов

Плагин `CustomStorage` предоставляет аналогичный функционал, что и работа с атрибутами лидов через `lead.setAttr()` и `lead.getAttr()` и т.д., но с дополнительными возможностями:

Функция	lead	customStorage
Работа с обычными атрибутами		
Работа с JSON атрибутами		
Точечная нотация для JSON		
Подкатегории (key2, key3)		
Работа с кастомными таблицами		
Прямая работа с БД		

Заключение

Плагин `CustomStorage` предоставляет мощный инструмент для работы с кастомными таблицами атрибутов в JavaScript скриптах. Он позволяет гибко хранить и получать данные с

поддержкой подкатегорий и вложенных JSON структур, что делает его незаменимым инструментом для сложных сценариев работы с данными.

Версия #4
metadock создал 28 December 2025 21:12:14
metadock обновил 27 January 2026 21:12:38