BIV

BIVPlaform

Руководство разработчика по компоненте Server Driven User Interface (SDI)

2025

Server-Driven User Interface

Общие понятия о SDUI

SDUI = SDI = Server Driven User Interface Другое наименование — BDUI = Backend Driven User Interface.

Назначение: формирование содержимого страницы пользовательского интерфейса на стороне бэкенда с помощью разметки и отображение этого содержимого на фронтенде.

- Нет, это не HTML. Это разметка более высокого уровня. Эта разметка не содержит специфических деталей реализации для конкретной платформы на стороне фронтенда. Фронтендом могут выступать не только веб, но и мобильные приложения или толстые десктопные клиенты. Фронетенд знает как отрисовывать каждую компоненту нативным способом. Стилистика элементов пользовательского интерфейса определяется на стороне фронтенда при создании каркаса приложения.
- Да, развитие технологий идёт по спирали. HTML —> SPA —> SDI — …? Разметка создаётся в формате JSON. Разметка достаточно простая, с её разработкой справится аналитик.

На текущий момент разметка включает в себя наиболее распространённые элементы пользовательского интерфейса. В ходе развития данная коллекция элементов будет пополняться.

Разметка SDUI

Содержимое страницы пользовательского интерфейса формируется в виде JSON.

Ниже приведён пример страницы с таблицей марок транспортных средств, из которой можно вызывать форму редактирования отдельной записи.

```
{
  "components": [
    {
      "type": "header",
      "caption": "Марка ТС"
    },
    {
      "type": "toolbar",
      "tools": [
        {
          "type": "button",
          "display": "minifab",
          "icon": "add",
          "color": "primary",
          "click": {
            "navigate": "/ins-ref/entity/vehicle-brand/table/form",
            "params": {
              "action": "create"
            }
          }
        },
        {
          "type": "spacer"
        },
        {
          "type": "button",
```

```
          "display": "icon",
          "icon": "refresh",
          "click": {
            "call": "VehicleBrandTable.reload"
          }
        },
        {
          "type": "button",
          "display": "icon",
          "icon": "search",
          "click": {
            "call": "VehicleBrandTable.filter"
          }
        },
        {
          "type": "button",
          "display": "icon",
          "icon": "settings",
          "click": {
            "call": "VehicleBrandTable.tools"
          }
        }
      ]
    },
    {
      "type": "journal",
      "id": "VehicleBrandTable",
      "entity": "VehicleBrand",
      "grid": "main",
      "uri": "/sdi/api/sdi-sandbox",
      "contextmenu": [
        {
          "text": "Редактировать",
          "icon": "edit",
          "click": {
            "navigate": "/ins-ref/entity/vehicle-brand/table/form",
            "params": {
              "action": "update",
              "id": "{__object_id}"
            }
          }
        },
        {
          "text": "Удалить",
          "icon": "delete",
          "click": {
            "navigate": "/ins-ref/entity/vehicle-brand/table/form",
            "params": {
              "action": "delete",
              "id": "{__object_id}"
            }
          }
        }
      ],
      "rowdblclick": {
        "navigate": "/ins-ref/entity/vehicle-brand/table/form",
        "params": {
          "action": "update",
          "id": "{__object_id}"
        }
      }
    }
  ]
}
```

Разберём основные элементы:

- “**components**” - список всех компонент, которые располагаются на странице. Каждая компонента имеет обязательное поле **type**, указывающее тип это компоненты. Например, “**type**”**:** “**header**”, т.е. заголовок страницы. Далее для компоненты указывается набор полей, специфичных для определённого типа компонент. Например, компонента типа “header” имеет поле “**caption**”, в котором указывается значение заголовка страницы.
- компонента **header** - заголовок страницы.
- компонента **toolbar** - панель инструментов с кнопками. В поле “**tools**” должны быть перечислены кнопки.
- компонента **button** - кнопка.
- компонента **spacer** - “заполнитель” пустого места. Специальный невидимый элемент, который заполняет пустое пространство в рамках строки и не даёт “растягиваться” другим элементам, располагающимся в этой же строке.
- компонента **journal** - гибкая таблица. Представление данных списоком в гриде с возможностью настройки колонок, сортировки, фильтров. Более подробно см. описание компонент.

Способы формирования разметки

Разметку SDI JSON можно софрмировать следующими способами:

1. “Вручную”. Т.е. аналитик/разработчик/кто-то пишет SDI JSON, который в дальнейшем будет выполняться.
2. Сгенераровать автоматически по описанию сущности. Для описания используется библиотека entity-info, которая используется и для гибких таблиц. По описанию класса @Entity будет сформирован полноценный пользовательский интерфейс с таблицей на базе «гибких таблиц» и формой редактирования. Класс @Entity должен включать в себя аннотации @EntityInfo и @AttributeInfo. В рамках @AttributeInfo можно описать валидаторы, которые будут отрабатывать при вводе данных.

Автоматическая генерация разметки SDI JSON

При автоматической генерации разметки SDI JSON есть возможность влиять на эту генерацию, а именно:

- изменять порядок следования элементов пользовательского интерфейса (контролов),
- изменять расположение контролов, например расположить несколько контролов в одной строке, для строк переопределять тип контрола, а именно сделать многострочный ввод вместо однострочного для поля типа примечание,
- изменять отдельные свойства контролов, например цвет / стиль кнопок.

Такая кастомизация делается таким же форматом разметки SDI JSON, но требуется указать только переопределяемые свойства.

При необходимости можно совместно использовать страницы, созданные «вручную» и сгенерированные автоматически.

Приложения SDI

С использованием данного механизма SDI можно создать два типа приложений:

1. Статическое приложение. SDI JSON разметка находится в файлах, которые являются ресурсами микросервиса. Структура веб-приложения определяется в ходе разработки микросервиса (как сейчас при традиционной разработке). Такой сервис работает «из коробки» и не требует что-либо ещё. После внесения изменений в разметку SDI JSON необходимо пересобрать микросервис.
2. Динамическое приложение. SDI JSON разметка находится в отдельной таблице в базе данных. Структура веб-приложения может дополняться «на лету» во время работы сервиса. Для работы сервиса необходима отдельной таблице sdi_page в базе данных. Все изменения в разметке SDI JSON подхватятся при следующем обращении к странице.

Механизм SDI реализован в виде набора библиотек и подключается к любому сервису на java/quarkus.

Более подробно см. раздел **Подключение на backend**

Подключение SDI на backend

Подключение библиотек с использованием maven

Для использования механизма SDI необходимо подключить несколько библиотек в зависимости от требуемого функционала (“ручной” JSON и/или автоматическая генерация) и типа приложения (статическое или/и динамическое получение страниц SDI JSON).

Подключение базовой библиотеки:

```
<dependency>
    <groupId>com.bivgroup.lib</groupId>
    <artifactId>sdi-base</artifactId>
    <version>${sdi.version}</version>
</dependency>
```

где ${sdi.version} - актуальная версия, на текущий момент 1.2-SNAPSHOT.

“Ручной” JSON и автоматическая генерация могут работать совместно или по отдельности.

“Ручной” JSON:

```
<dependency>
    <groupId>com.bivgroup.lib</groupId>
    <artifactId>sdi-static</artifactId>
    <version>${sdi.version}</version>
</dependency>
```

Автоматическая генерация по описанию сущности:

```
<dependency>
    <groupId>com.bivgroup.lib</groupId>
    <artifactId>sdi-plain-entity</artifactId>
    <version>1.0-SNAPSHOT</version>
</dependency>
```

В дополнение к библиотеке выше непосредственно для самого описания сущностей и работы с ними необходимо подключить:

```
<dependency>
    <groupId>com.bivgroup.lib</groupId>
```

```
            <artifactId>entity-info</artifactId>
            <version>${entity-journal.version}</version>
        </dependency>
        <dependency>
            <groupId>com.bivgroup.lib</groupId>
            <artifactId>entity-data</artifactId>
            <version>${entity-journal.version}</version>
        </dependency>
        <dependency>
            <groupId>com.bivgroup.lib</groupId>
            <artifactId>entity-journal</artifactId>
            <version>${entity-journal.version}</version>
        </dependency>
```

где ${entity-journal.version} - актуальная совместимая версия entity-journal, на текущий момент 3.11-SNAPSHOT.

Для entity-journal необходимо подключить реализацию интерфейса авторизации. Более подробно см. документацию по entity-journal - **Авторизация (права)**

Вот пример подключения тестовой реалиции интерфейса авторизации:

```
        <dependency>
            <groupId>com.bivgroup.lib</groupId>
            <artifactId>entity-journal-test-auth</artifactId>
            <version>${entity-journal.version}</version>
        </dependency>
```

Для корректтной работы расширения entity-info требуется подключение плагина jandex-maven-plugin:

```
    <build>
        <plugins>
            <plugin>
                <groupId>org.jboss.jandex</groupId>
                <artifactId>jandex-maven-plugin</artifactId>
                <version>${jandex-maven-plugin.version}</version>
                <executions>
                    <execution>
                        <id>make-index</id>
                        <goals>
                            <goal>jandex</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
```

Требуется выбрать тип получения страниц с SDI JSON:

- “статическое” - вся SDI JSON разметка находится в ресурсах quarkus-приложения в файлах. “Статический” SDI JSON НЕ может быть изменён в runtime.
- “динамическое” - вся SDI JSON разметка находится в таблице в базе данных. “Динамический” SDI JSON может быть модифицирован в runtime, созданы новые страницы или удалены существующие.

Для статического получения SDI JSON:

```
        <dependency>
            <groupId>com.bivgroup.lib</groupId>
            <artifactId>sdi-file-storage</artifactId>
            <version>${sdi.version}</version>
        </dependency>
```

Для динамическоого получения SDI JSON помимо основной библиотеки **sdi-db-storage** требуется подключить реализацию интерфейса авторизации, например **sdi-db-storage-auth-test**.

```
        <dependency>
            <groupId>com.bivgroup.lib</groupId>
            <artifactId>sdi-db-storage</artifactId>
            <version>${sdi.version}</version>
        </dependency>
        <dependency>
            <groupId>com.bivgroup.lib</groupId>
            <artifactId>sdi-db-storage-auth-test</artifactId>
            <version>${sdi.version}</version>
        </dependency>
```

Возможно одновременное подключение “статических” и “динамических” страниц, при этом приоритет имеют “динамические” страницы, считанные из БД.

Разметка SDI JSON

Подключение в проект

Описание

Компонент-обёртка для встраивания интерпретатора SDI

Свойства

Наименование	Тип	Обязательность	По умолчанию	Описание
`instruction`	`SDIPage | undefined`	Нет	`undefined`	Инструкция интерпритатора (two-way binding)
`observeRouting`	`boolean`	Нет	`false`	Отслеживать изменения роутинга приложения и запрашивать инструкцию с бэка (см. `SDI_API_PREFIX`)

Настройка

Токен `SDI_API_PREFIX`

Установка префикса для запросов на сервер

```
import { PlatformLocation } from '@angular/common';
import { SDI_API_PREFIX } from '@biv/sdi';

export const appConfig: ApplicationConfig = {
    providers: [
        {
            provide: SDI_API_PREFIX,
```

```
            useFactory: (location: PlatformLocation) =>
location.getBaseHrefFromDOM().replace(/\/+$/, "") + "/api/sdi-sandbox",
            deps: [PlatformLocation],
        },
    ]
}
```

Примеры

Простая страница со статической инструкцией

```
import { Component } from '@angular/core';
import { SdiModule } from '@biv/sdi';

@Component({
    template: '<sdi-wrapper [instruction]="instruction" />',
    standalone: true,
    imports: [SdiModule],
})
export class SdiPageComponent {

    public readonly instruction = {
        components: [
            {
                type: "header",
                caption: "Hello SDI!"
            }
        ]
    };

}
```

Пример компонента с режимом отслеживания роутинга приложения

Настройка роутинга приложения `src/app/app.routes.ts`

```
import { Routes } from '@angular/router';

import { SomePageComponent } from './some-page/some-page.component';
import { SdiPageComponent } from './sdi-page/sdi-page.component';


export const routes: Routes = [
    { path: "some-page", component: SomePageComponent },
    { path: "**", component: SdiPageComponent }, // Все неописанные пути
обрабатываются компонентом SdiPageComponent
];
```

Код SDI компонента `src/app/sdi-page/sdi-page.component`

```
import { Component } from '@angular/core';
import { SdiModule } from '@biv/sdi';

@Component({
    template: '<sdi-wrapper observeRouting />',
    standalone: true,
    imports: [SdiModule],
})
export class SdiPageComponent { }
```

Компоненты SDI

actionbar

Описание

Данный компонент используется для отображения кнопок в горизонтальной панели.

> При прокрутке, данный элемент прилипает к нижнему краю экрана.
> {.is-info}

Свойства

Наименование	Тип	Обязательность	Описание
type	string	Да	Тип компонента (`actionbar`)
buttons	Array< Button | Spacer >	Да	Массив вложенных кнопок или разделителей

Поддерживаемые реакции эффектов

Реакция	Тип	Описание
hidden	boolean	Скрыть панель (если `true`)

Примеры

Форма с действиями

```
{
    "components": [
        {
            "type": "container",
            "components": [
                {
                    "id": "MyForm",
                    "type": "form",
                    "cols": 2,
                    "fields": [
                        {
                            "type": "integer",
                            "name": "id",
                            "hidden": true
                        },
                        {
                            "type": "string",
                            "name": "name",
                            "label": "Наименование",
                            "span": 2,
                            "validators": {
                                "required": true
                            }
                        }
                    ]
                },
                {
```

```
                    "type": "actionbar",
                    "buttons": [
                        {
                            "type": "spacer"
                        },
                        {
                            "type": "button",
                            "text": "Отменить",
                            "click": {
                                "navigate": "/some/path"
                            }
                        },
                        {
                            "type": "button",
                            "text": "Изменить",
                            "icon": "edit",
                            "display": "flat",
                            "color": "primary",
                            "click": {
                                "call": "MyForm.validate"
                            }
                        }
                    ]
                }
            ]
        }
    ]
}
```

button

Описание

Данный компонент используется для отображения кнопок.

> Для отображения используется mat-button.
> {.is-info}

Свойства

Наименование	Тип	Обязательность	Описание	Пример
type	string	Да	Тип компонента	`"type": "button"`
text	string	Нет	Текст	`"text": "Сохранить"`
icon	string	Нет	Иконка	`"icon": "add"`
color	`primary` | `accent` | `warn`	Нет	Цвет	`"color": "primary"`
display	`raised` | `stroked` | `flat` |	Нет	Стиль отображения	`"display": "flat"`

Наименование	Тип	Обязательность	Описание	Пример
	`icon` | `fab` | `mini fab`			
click	Action	Нет	Описание действия при нажатии на кнопку	`"click": { "navigate": "/some/page" }`

Т.к. в качестве компонентов используется Angular Material, то для иконок используется Google Fonts. {.is-info}

Поддерживаемые реакции эффектов

Реакция	Тип	Описание
text	string	Текст кнопки
icon	string	Название иконки
color	`primary` | `accent` | `warn`	Цвет
display	`raised` | `stroked` | `flat` | `icon` | `fab` | `minifab`	Стиль отображения
disabled	boolean	Деактивировать кнопку (если `true`)
hidden	boolean	Скрыть кнопку (если `true`)

Пример

```
{
  "data$": {
    "myButton": null
  },
  "components": [
    {
      "type": "form",
      "data$": "myButton",
      "cols": 2,
      "fields": [
        {
          "type": "select",
          "name": "display",
          "label": "Представление",
          "span": 2,
          "options": [
            {
              "objectName": "-- Не определено --"
            },
            {
              "id": "raised",
              "objectName": "raised"
            },
            {
              "id": "stroked",
              "objectName": "stroked"
            },
```

```
    {
      "id": "flat",
      "objectName": "flat"
    },
    {
      "id": "fab",
      "objectName": "fab"
    },
    {
      "id": "minifab",
      "objectName": "minifab"
    }
  ]
},
{
  "type": "select",
  "name": "color",
  "label": "Цвет",
  "span": 2,
  "options": [
    {
      "objectName": "-- Не выбран --"
    },
    {
      "id": "primary",
      "objectName": "primary"
    },
    {
      "id": "accent",
      "objectName": "accent"
    },
    {
      "id": "warn",
      "objectName": "warn"
    }
  ]
},
{
  "type": "boolean",
  "name": "noText",
  "label": "Без подписи",
  "span": 2,
  "value": false
},
{
  "type": "string",
  "name": "text",
  "label": "Подпись",
  "span": 2,
  "value": "Кнопка",
  "effect": {
    "if": {
      "operator": "AND",
      "predicates": [
        {
          "attr": "myButton.noText",
          "operator": "EQUAL",
          "value": true
        }
      ]
    },
    "then": {
      "disabled": true
    },
```

```
            "else": {
              "disabled": false
            }
          }
        },
        {
          "type": "boolean",
          "name": "noIcon",
          "label": "Без иконки",
          "span": 2,
          "value": false
        },
        {
          "type": "string",
          "name": "icon",
          "label": "Иконка",
          "span": 2,
          "value": "add",
          "effect": {
            "if": {
              "operator": "AND",
              "predicates": [
                {
                  "attr": "myButton.noIcon",
                  "operator": "EQUAL",
                  "value": true
                }
              ]
            },
            "then": {
              "disabled": true
            },
            "else": {
              "disabled": false
            }
          }
        },
        {
          "type": "content",
          "html": "<a href=\"https://fonts.google.com/icons\"
target=\"_blank\">Коллекция иконок</a>"
        },
        {
          "type": "boolean",
          "name": "disabled",
          "label": "Сделать кнопку не активной",
          "span": 2,
          "value": false
        },
        {
          "type": "boolean",
          "name": "hidden",
          "label": "Скрыть кнопку",
          "span": 2,
          "value": false
        }
      ]
    },
    {
      "type": "actionbar",
      "buttons": [
        {
          "type": "button",
          "text": "Кнопка 3",
```

```
"effect": [
  {
    "if": {
      "operator": "AND",
      "predicates": [
        {
          "attr": "myButton.display",
          "operator": "IS_NOT_NULL"
        }
      ]
    },
    "then": {
      "display": "${myButton.display}"
    },
    "else": {
      "display": null
    }
  },
  {
    "if": {
      "operator": "AND",
      "predicates": [
        {
          "attr": "myButton.color",
          "operator": "IS_NOT_NULL"
        }
      ]
    },
    "then": {
      "color": "${myButton.color}"
    },
    "else": {
      "color": null
    }
  },
  {
    "if": {
      "operator": "AND",
      "predicates": [
        {
          "attr": "myButton.noText",
          "operator": "EQUAL",
          "value": true
        }
      ]
    },
    "then": {
      "text": null
    }
  },
  {
    "if": {
      "operator": "AND",
      "predicates": [
        {
          "attr": "myButton.text",
          "operator": "IS_NOT_NULL"
        }
      ]
    },
    "then": {
      "text": "${myButton.text}"
    }
  },
```

```
{
  "if": {
    "operator": "AND",
    "predicates": [
      {
        "attr": "myButton.noIcon",
        "operator": "EQUAL",
        "value": true
      }
    ]
  },
  "then": {
    "icon": null
  }
},
{
  "if": {
    "operator": "AND",
    "predicates": [
      {
        "attr": "myButton.icon",
        "operator": "IS_NOT_NULL"
      }
    ]
  },
  "then": {
    "icon": "${myButton.icon}"
  }
},
{
  "if": {
    "operator": "AND",
    "predicates": [
      {
        "attr": "myButton.disabled",
        "operator": "EQUAL",
        "value": true
      }
    ]
  },
  "then": {
    "disabled": true
  },
  "else": {
    "disabled": false
  }
},
{
  "if": {
    "operator": "AND",
    "predicates": [
      {
        "attr": "myButton.hidden",
        "operator": "EQUAL",
        "value": true
      }
    ]
  },
  "then": {
    "hidden": true
  },
  "else": {
    "hidden": false
  }
```

```
            }
          ]
        }
      ]
    }
  ]
}
```

container

Описание

Данный компонент используется объединения компонентов в контейнер.

Для обёркти используется mat-card, где используется только `mat-card-content` для встраивания контента.

Свойства

Наименивание	Тип	Обязательность	Описание	Пример
type	string	Да	Тип компонента	`"type": "container"`
components	Array<component | container>	Да	Массив вложенных компонентов или контейнеров	`"components": [ { "type": "button", "text": "Кнопка" } ]`

Поддерживаемые реакции эффектов

Реакция	Тип	Описание
hidden	boolean	Скрыть панель (если `true`)

content

Описание

Данный компонент используется объединения компонентов в контейнер.

Для обёркти используется mat-card, где используется только `mat-card-content` для встраивания контента.

Свойства

Наименивание	Тип	Обязательность	Описание	Пример
type	string	Да	Тип компонента	`"type": "container"`
components	Array<component | container>	Да	Массив вложенных компонентов или контейнеров	`"components": [ { "type": "button", "text": "Кнопка" }`

Наименование	Тип	Обязательность	Описание	Пример]

Поддерживаемые реакции эффектов

Реакция	Тип	Описание
hidden	boolean	Скрыть панель (если `true`)

divider

Описание

Данный компонент используется для отображения вертикального или горизонтального разделителя компонентов.
Он поддерживается компонентами: `toolbar`, `actionbar`, `menu` и в общем списке компонентов.
В качестве компонента используется <mat-divider>

Свойства

Наименование	Тип	Обязательность	Описание
type	string	Да	Тип компонента (`divider`)
vertical	boolean	Нет	Отображение вертикальной черты в горизонтальных списках
inset	boolean	Нет	Отображается с отступом слева в горизонтальных списках

Поддерживаемые реакции эффектов

Реакция	Тип	Описание
hidden	boolean	Скрыть панель (если `true`)

Пример

Разделитель в панели инструментов (с отступом)

```
{
    "components": [
        {
            "type": "toolbar",
            "tools": [
                {
                    "type": "header",
                    "caption": "Заголовок страницы"
                },
                {
                    "type": "divider",
                    "vertical": true,
                    "inset": true
                },
                {
                    "type": "button",
```

```
                    "display": "raised",
                    "icon": "add",
                    "text": "Создать"
                }
            ]
        }
    ]
}
```

Разделитель в actionbar (без отступа)

```
{
    "components": [
        {
            "type": "actionbar",
            "buttons": [
                {
                    "type": "button",
                    "display": "flat",
                    "color": "primary",
                    "icon": "add",
                    "text": "Создать"
                },
                {
                    "type": "divider",
                    "vertical": true
                },
                {
                    "type": "button",
                    "display": "flat",
                    "color": "accent",
                    "icon": "delete",
                    "text": "Удалить"
                }
            ]
        }
    ]
}
```

Разделитель в выпадающем меню

```
{
    "components": [
        {
            "type": "toolbar",
            "tools": [
                {
                    "type": "menu",
                    "items": [
                        {
                            "icon": "home",
                            "text": "Меню",
                            "items": [
                                {
                                    "text": "Пункт меню 1",
                                    "icon": "add"
                                },
                                {
                                    "type": "divider"
                                },
                                {
                                    "text": "Пункт меню 2",
                                    "icon": "edit"
                                }
```

```
                        ]
                    }
                ]
            }
        ]
    }
]
}
```

form

Описание

Данный компонент используется для отображения форм редактирования.
В разделе **элементы формы** можно посмотреть дополнительную информацию по полям.

Свойства

Наименование	**Тип**	**Обязательность**	**Описание**	**Пример**
type	string	Да	Тип компонента	`"type": "form"`
id	string	Нет	Уникальный идентификатор компонента	`"id": "MyForm"`
api	Api	Нет	Описание API формы	`"api": { "read": { "method": "GET", "uri": "/api/v1/data/service1/{id}", "response": "self" } }`
onInit	Action	Нет	Действие при инициализации формы	`"onInit": { "call": "self.read" }`
fields	Array<FormField|FormElement>	Да	Список полей формы	`"fields": [ { "type": "string", "name": "field1", "label": "Поле 1" } ]`
cols	number	Нет	Кол-во колонок в форме	`"cols": "4"`
expandable	boolean	Нет	Сделать формы сворачиваемой	`"expandable": "true"`

Размер колонки в форме задан как 10rem справка по rem, em.
Т.е. инструкция вида `{ "type": "form", "cols": 4 }` отрисует форму шириной в 40rem (640px). {.is-info}

Встроенные методы

Наименование	**Описание**
validate	Заппустить механизм проверки коррекнтности заполнения полей формы и подсветить ошибки

Поддерживаемые реакции эффектов

Реакция	**Тип**	**Описание**
hidden	boolean	Скрыть панель (если `true`)

Примеры

Простая форма с полями

```
{
    "components": [
        {
            "type": "container",
            "components": [
                {
                    "type": "form",
                    "cols": 3,
                    "fields": [
                        {
                            "type": "integer",
                            "name": "id",
                            "hidden": true
                        },
                        {
                            "type": "string",
                            "name": "field1",
                            "label": "Поле типа \"string\"",
                            "span": 3
                        },
                        {
                            "type": "divider"
                        },
                        {
                            "type": "integer",
                            "name": "field2",
                            "label": "Поле типа \"integer\""
                        },
                        {
                            "type": "decimal",
                            "name": "field3",
                            "precision": 3,
                            "label": "Поле типа \"decimal.000\"",
                            "value": 999.99
                        },
                        {
                            "type": "divider"
                        },
                        {
                            "type": "boolean",
                            "name": "field4",
                            "label": "Поле типа \"boolean\""
                        },
                        {
                            "type": "date",
                            "name": "field5",
```

```
                        "label": "Поле типа \"date\""
                    },
                    {
                        "type": "select",
                        "name": "field6",
                        "label": "Поле типа \"select\"",
                        "options": [
                            {
                                "id": null,
                                "objectName": "-- выберите значение --"
                            },
                            {
                                "id": 1,
                                "objectName": "Выбор 1"
                            }
                        ]
                    },
                    {
                        "type": "textarea",
                        "name": "field7",
                        "label": "Поле типа \"textarea\"",
                        "span": 3
                    }
                ]
            }
        ]
    }
]
}
```

Заполнение формы данными из указанного источника

```
{
    "components": [
        {
            "type": "form",
            "onInit": {
                "call": "self.read"
            },
            "api": {
                "read": {
                    "method": "GET",
                    "uri": "/api/v1/data/service1/{id}",
                    "response": "self"
                }
            },
            "fields": [
                {
                    "type": "integer",
                    "name": "id",
                    "hidden": true
                },
                {
                    "type": "string",
                    "name": "field1",
                    "label": "Поле типа \"string\""
                }
            ]
        }
    ]
}
```

После отрисовки формы, интерпретатор SDI отрабатывает действие прописанное в атрибуте `onInit`.

В данном примере вызывается метод `read`, описанный в секции `api`.

> Значение `self` ссылается на компонент, в котором описано действие, т.е. на компонент формы. {.is-info}
>
> Значение `self`, указанное в блоке `api -> read -> response`, говорит интерпретатору, взять тело ответа и поместить его в форму. {.is-info}
>
> Параметр `{id}` в атрибуте `api -> read -> uri` подставляется из query-параметров страницы. {.is-info}
>
> **Важно!**
> Если в адресе страницы отсутсвует query-параметр `id`, то действие, описанное в блоке `onInit`, не отработает.
> Это является условностью данного интерпретатора. {.is-warning}

Валидация и сохранение формы

```
{
    "components": [
        {
            "type": "form",
            "id": "MyForm",
            "onInit": {
                "call": "self.read"
            },
            "api": {
                "read": {
                    "method": "GET",
                    "uri": "/api/v1/data/service1/{id}",
                    "response": "self"
                },
                "save": {
                    "method": "POST",
                    "uri": "/api/v1/service/entity/{id}",
                    "request": "self"
                }
            },
            "cols": 2,
            "fields": [
                {
                    "type": "integer",
                    "name": "id",
                    "hidden": true
                },
                {
                    "type": "string",
                    "name": "field1",
                    "label": "Поле типа \"string\"",
                    "span": 2,
                    "validators": {
                        "required": true
                    }
                }
            ]
        },
        {
            "type": "actionbar",
            "buttons": [
                {
```

```
                    "type": "button",
                    "text": "Сохранить",
                    "click": {
                        "call": "MyForm.validate",
                        "chain": {
                            "call": "MyForm.save"
                        }
                    }
                }
            ]
        }
    ]
}
```

В данном примере секция `api` дополнена блоком `save`, в котором описан вызов back-end сервиса.
В компонент формы добавлен `"id": "MyForm"` - уникальный идентификатор компонента формы на странице.
Добавлен дополнительный компонент `actionbar` с кнопкой `"Сохранить"`.
В кнопке указана цепочка действий:

1. `"call": "MyForm.validate"` - вызывает встроенный в компонент метод `validate`, который вызывает механизм проверки корректной заполненности полей. Если формы заполнена корректно, то вызывается следующее действие, описанное в блоке `"chain"`.
2. `"call": "MyForm.save"` - вызывает метод `save`, описанный в блоке `api` копмонента формы.

   Значение `self`, указанное в блоке `api -> save -> request`, говорит интерпретатору, взять значения формы и передать его в теле запроса. {.is-info}

Элементы формы

Описание

В данном разделе описаны элементы, которые являются частью формы.

Общие свойства полей формы

Параметр	Тип	Обязательность	Описание
type	string	Да	Тип поля
name	string	Да	Системное имя поля (атрибута)
label	string	Нет	Лэйюл (наименование, выводимое на интерфейсе)
value	any	Нет	Значение, устанавливаемое в поле, при инициализации
hidden	boolean	Нет	Скрытое поле (поле не отображается в форме, но передаётся в сохраняемом объекте формы)
readonly	boolean	Нет	Неизменяемое поле

Параметр	Тип	Обязательность	Описание
			(поле отображается в форме и передаётся в сохраняемом объекте формы, но его значение нельзя изменить)
disabled	boolean	Нет	Неактивное поле (поле не передаётся в сохраняемом объекте формы)
span	number	Нет	Кол-во занимаемых колонок в строке (см. `form.cols`)
prefix	string	Нет	Статичное значение, отображаемое в начале поля (не применяется к значению поля)
suffix	string	Нет	Статичное значение, отображаемое в конце поля (не применяется к значению поля)
validators	Validators	Нет	Описание валидаторов для поля
effect	Array<Effect> | Effect	Нет	Описание эффектов, применяемых к полю

Типы полей

string

Поле для ввода строковых значений

```
{
    "type": "string",
    "name": "lastName",
    "label": "Фамилия"
}
```

Валидаторы `min` и `max` проверяют кол-во символов в строке
{.is-info}

Валидатор `pattern` проверяют строку на соответствие указанному регулярному выражению
{.is-info}

textarea

Поле для ввода многострочного текста

```
{
    "type": "textarea",
```

```
    "name": "comment",
    "label": "Комментарий"
}
```

Валидаторы `min` и `max` проверяют кол-во символов в строке
{.is-info}

Валидатор `pattern` проверяют строку на соответствие указанному регулярному выражению
{.is-info}

boolean

Поле для ввода булевого значение

```
{
    "type": "boolean",
    "name": "hasChildren",
    "label": "Есть дети"
}
```

date

Поле для ввода значения типа дата

```
{
    "type": "date",
    "name": "beginDate",
    "label": "Дата начала"
}
```

Валидаторы `min` и `max` проверяют дату на указанные ограничения
{.is-info}

integer

Поле для ввода целых чисел

```
{
    "type": "integer",
    "name": "numberOfCars",
    "label": "Кол-во автомобилей"
}
```

Валидаторы `min` и `max` проверяют число на указанные ограничения
{.is-info}

Дополнительные свойства поля

Параметр	Тип	Обязательность	Описание
asIs	boolean	Нет	Отображаемое значение как есть, без форматирования

decimal, bigdecimal, float

Поле для ввода чисел с плавающей запятой

```
{
    "type": "decimal",
    "name": "price",
```

```
    "label": "Стоимость"
}
```

Валидаторы `min` и `max` проверяют число на указанные ограничения
{.is-info}

Дополнительные свойства поля

Параметр	Тип	Обязательность	Описание
precision	number	Нет	Отображаемое кол-во знаков после запятой
asIs	boolean	Нет	Отображаемое значение как есть, без форматирования

Во избежании округления значений, вида `999999.99`, браузером, используется библиотека bignumber.js.
Все значения на бэк передаются в виде строки.
{.is-warning}

select

Поле для выбора значения из списка (со статичными значениями)

```
{
    "type": "select",
    "name": "animal",
    "label": "Домашнее животное",
    "options": [
        { "id": null, "objectName": "--- Пустое значение ---" },
        { "id": "cat", "objectName": "Кошка" },
        { "id": "dog", "objectName": "Собака" }
    ]
}
```

Поле для выбора значения из списка (со значениями полученными от backend-сервиса)

```
{
    "type": "select",
    "name": "animal",
    "label": "Домашнее животное",
    "api": {
        "uri": "/api/v1/animal/list"
    }
}
```

Дополнительные свойства поля

Параметр	Тип	Обязательность	Описание
api	Api	Нет	API для заполнение селектора доступными значениями
options	Array<FormFieldSelectOption>	Нет	Статичный список значений селектора

FormFieldSelectOption

Параметр	Тип	Обязательность	Описание
id	string |	Да	Подставляемое

Параметр	Тип	Обязательность	Описание
	number | null		значение в поле типа `select`
objectName	string	Да	Отображаемое значение в поле типа `select`

autocomplete

Поле для выбора значения из списка (со динамическими значениями по пользовательскому вводу)
В ответе бэк должен прислыать объект `FormFieldSelectOption`

```
{
    "name": "brandId",
    "type": "autocomplete",
    "label": "Марка ТС",
    "minQuery": 2,
    "api": {
        "method": "POST",
        "uri": "/data/autocomplete",
        "payload": {
            "entityBrief": "VehicleBrand",
            "attr": "nameEng",
            "maxRows": 10
        }
    }
}
```

Дополнительные свойства поля

Параметр	Тип	Обязательность	Описание
api	Api	Нет	API для заполнение селектора доступными значениями
minQuery	number	Нет	Минимальное кол-во введёных пользователем символов, для запроса на бэк

lookup

Поле для выбора значения из справочника (журнал в модальном окне)

```
{
    "type": "lookup",
    "name": "animal",
    "label": "Домашнее животное",
    "uri": "/sdi/api/animal/choice",
    "entity": "animal",
    "grid": "lookup",
    "dataLoader": {
        "method": "GET",
        "uri": "/data/crud/animal/{__object_id}"
    },
}
```

Дополнительные свойства поля

Параметр	Тип	Обязательность	Описание
uri	string	Да	End-point журнала
entity	string	Да	Наименование сущности журнала
grid	string	Да	Наименование группы настроек журнала
dataLoader	Api	Да	Разыменование значения
filterBy	JournalFilter	Нет	Установить фильтр для вызываемого журнала

group

Поле для ввода значений в подгруппу

```
{
    "type": "group",
    "name": "child",
    "label": "Ребёнок",
    "fields": [
        { "type": "string", "name": "lastName", "label": "Фамилия" },
        { "type": "string", "name": "firstName", "label": "Имя" },
        { "type": "string", "name": "secondName", "label": "Отчество" }
    ],
    "expandable": true
}
```

Будет сформирован документ вида

```
{
    "child": {
        "lastName": "Иванов",
        "firstName": "Иван",
        "secondName": "Иванович"
    }
}
```

Дополнительные свойства поля

Параметр	Тип	Обязательность	Описание
fields	Array<FormField>	Да	Массив вложенных полей
expandable	boolean	Нет	Установть группу полей как раскрываемую

array

Поле для ввода значений в виде массива

```
{
    "type": "array",
    "name": "children",
    "label": "Дети",
    "fields": [
        { "type": "string", "name": "lastName", "label": "Фамилия" },
        { "type": "string", "name": "firstName", "label": "Имя" },
        { "type": "string", "name": "secondName", "label": "Отчество" }
```

```
    ],
    "expandable": true
}
```

Будет сформирован документ вида

```
{
    "children": [
        {
            "lastName": "Иванов",
            "firstName": "Иван",
            "secondName": "Иванович"
        },
        {
            "lastName": "Петров",
            "firstName": "Пётр",
            "secondName": "Петрович"
        },
        {
            "lastName": "Александров",
            "firstName": "Александр",
            "secondName": "Александрович"
        }
    ]
}
```

Дополнительные свойства поля

Параметр	Тип	Обязательность	Описание
fields	Array<FormField>	Да	Массив вложенных полей
expandable	boolean	Нет	Установть группу полей как раскрываемую
itemCols	number	Нет	Кол-во колонок для вложенных элементов

fieldset

Поле для визуального объединения полей в подгруппу. Обычно используется для применения эффектов к группе полей.

```
{
    "type": "fieldset",
    "fields": [
        { "type": "string", "name": "name", "label": "Наименование" },
        { "type": "decimal", "name": "price", "label": "Стоимость" },
        { "type": "boolean", "name": "favorites", "label": "Избранное" }
    ]
}
```

Будет сформирован документ вида

```
{
    "name": "Какая-то вещь",
    "price": 9999.99,
    "favorites": true
}
```

Дополнительные свойства поля

Параметр	Тип	Обязательность	Описание
fields	Array<FormField>	Да	Массив группируемых полей

Параметр	Тип	Обязательность	Описание
expandable	boolean	Нет	Установть группу полей как раскрываемую

Данный тип поля не использует параметр `name`, т.к. используется исключительно для визуальной группировки или группового применения эффектов
{.is-warning}

Валидаторы

Инструкция для проверки валидности введённых данных в поле.

Наименование	Тип	Для полей типа	Описание
required	boolean | FieldValidatorExt	Все	Проверяет, заполнено поле или нет
min	number | FieldValidatorExt	Числовые, строковые и поля типа дата	Проверяет, на минимальное значение для чисел, минималдьное кол-во знаков для строк, минимальную дату для дат
max	number | FieldValidatorExt	Числовые, строковые и поля типа дата	Проверяет, на максимальное значение для чисел, максимальное кол-во знаков для строк, минимальную дату для дат
pattern	string | FieldValidatorExt	Строковые поля	Проверяет введёное значение на соответствие указанному регулярному выражению

FieldValidatorExt

Расширенное описание валидатора, можно указать собственное сообщение об ошибке.

Параметр	Тип	Обязательность	Описание
value	any	Да	Значение валидатора
text	string	Да	Сообщение об ошибке, выводимое в случае, если проверяемое значение не прошло

Параметр	Тип	Обязательность	Описание
			проверку

Примеры

```
{
    "components": [
        {
            "id": "MyForm1",
            "type": "form",
            "fields": [
                {
                    "type": "string",
                    "name": "lastName",
                    "label": "Фамилия",
                    "validators": {
                        "required": true,
                        "min": {
                            "value": 2,
                            "text": "Фамилия должна содержать не меньше 2-х
символов"
                        }
                    }
                },
                {
                    "type": "integer",
                    "name": "age",
                    "label": "Возраст",
                    "validators": {
                        "required": {
                            "value": true,
                            "text": "Необходимо указать свой возраст"
                        },
                        "min": 18,
                        "max": {
                            "value": 100,
                            "text": "Возраст не должен превышать 100 лет"
                        }
                    }
                },
                {
                    "type": "string",
                    "name": "policyNumber",
                    "label": "Номер полиса",
                    "validators": {
                        "required": true,
                        "pattern": {
                            "value": "[0-9]{3}-[A-Z]{3}",
                            "text": "Укажите номер полиса в формате \"999-AAA\""
                        }
                    }
                },
                {
                    "type": "button",
                    "display": "flat",
                    "icon": "save",
                    "color": "primary",
                    "text": "Submit",
                    "click": {
                        "call": "MyForm1.validate"
                    }
                }
            ]
```

```
        }
    ]
}
```

Дополнительные элементы формы

Дополнительные элементы не являются компонентами для ввода данных, а используются в декоративных или иных утилитарных целях.

spacer

Данный элемент используется для создания пустого пространства между элементами формы.

```
{
    "components": [
        {
            "id": "MyForm1",
            "type": "form",
            "cols": 3,
            "fields": [
                {
                    "type": "string",
                    "name": "lastName",
                    "label": "Фамилия"
                },
                {
                    "type": "spacer",
                    "span": 2
                },
                {
                    "type": "spacer",
                    "span": 1
                },
                {
                    "type": "string",
                    "name": "firstName",
                    "label": "Имя"
                },
                {
                    "type": "spacer",
                    "span": 1
                },
                {
                    "type": "spacer",
                    "span": 2
                },
                {
                    "type": "string",
                    "name": "secondName",
                    "label": "Отчество"
                }
            ]
        }
    ]
}
```

divider

Данный элемент используется для разделения компонентов формы горизонтальной чертой.

Данная черта занимает всю строчку и не имеет атрибута `span` {.is-info}

```
{
```

```
    "components": [
        {
            "type": "form",
            "cols": 3,
            "fields": [
                {
                    "type": "string",
                    "name": "lastName",
                    "label": "Фамилия"
                },
                {
                    "type": "string",
                    "name": "firstName",
                    "label": "Имя"
                },
                {
                    "type": "divider"
                },
                {
                    "type": "string",
                    "name": "secondName",
                    "label": "Отчество"
                }
            ]
        }
    ]
}
```

button

Данный элемент встраивает кнопку в сетку вместе с полями.

> Размеры кнопки меняются в соответствии с сеткой формы.
> {.is-info}

```
{
    "components": [
        {
            "id": "MyForm1",
            "type": "form",
            "cols": 2,
            "fields": [
                {
                    "type": "string",
                    "name": "lastName",
                    "label": "Фамилия",
                    "validators": {
                        "required": true
                    }
                },
                {
                    "type": "string",
                    "name": "firstName",
                    "label": "Имя",
                    "validators": {
                        "required": true
                    }
                },
                {
                    "type": "string",
                    "name": "secondName",
                    "label": "Отчество"
                },
```

```
                {
                    "type": "button",
                    "display": "flat",
                    "icon": "save",
                    "color": "primary",
                    "text": "Submit",
                    "click": {
                        "call": "MyForm1.validate"
                    }
                }
            ]
        }
    ]
}
```

content

Данный элемент встраивает текст / html в сетку формы.

```
{
    "components": [
        {
            "id": "MyForm1",
            "type": "form",
            "cols": 2,
            "fields": [
                {
                    "type": "string",
                    "name": "lastName",
                    "label": "Фамилия",
                    "validators": {
                        "required": true
                    }
                },
                {
                    "type": "string",
                    "name": "firstName",
                    "label": "Имя",
                    "validators": {
                        "required": true
                    }
                },
                {
                    "type": "content",
                    "text": "Отвество можно не указывать."
                },
                {
                    "type": "string",
                    "name": "secondName",
                    "label": "Отчество"
                }
            ]
        }
    ]
}
```

Эффекты

Эффекты используются, что бы описать взаимодействие полей формы между собой. В эффектах описывается условие, при котором на компонент формы оказывается воздействие.

Эффекты работают с источниками данных, убедитесь, что в инструкции описан источник и с ним связана форма
{.is-info}

Параметр	Тип	Обязательность	Описание
if	Condition	Да	Условие срабатывания эффекта
then	EffectReaction	Да	Действия при выполнении условия
else	EffectReaction	Нет	Действия при НЕ выполнении условий

EffectReaction

Параметр	Тип	Описание
calculate	string	Вычислить значение
evaluate	any	Присвоить значение
hidden	boolean	Скрыть / Показать элемент
disabled	boolean	Деактивировать / Активировать элемент
readonly	boolean	Запретить / Разрешить редактирование данных в элемемнте
prefix	string	Установить префикс для поля
suffix	string	Установить суффикс для поля
filterBy	JournalFilter	Установить фильтр для журнала (работает только для поля типа `lookup`)
sortBy	JournalSorting	**TODO:** Установить сортировку для журнала (работает только для поля типа `lookup`)

Примеры

Активация / Деактивация поля в форме

Поле “**Отчество**” становится активным, если поле “**Без отчества**” установлено в значение `false`
В инструкции описан источник данны `user` в блоке `data$`, в форме описана связь `"data$": "user"`
Условие в эффекте ссылается на источник данных `user` и отслеживает состояние атрибута `noSecondName`

Обратите внимание, что валидатор `required` для поля **secondName** отрабатывает только в том случае, если поле активно (`"disabled": false`)
{.is-info}

```
{
    "data$": {
        "user": null
    },
    "components": [
        {
            "type": "form",
```

```
        "id": "MyForm1",
        "data$": "user",
        "fields": [
            {
                "type": "string",
                "name": "lastName",
                "label": "Фамилия",
                "validators": {
                    "required": true
                }
            },
            {
                "type": "string",
                "name": "firstName",
                "label": "Имя",
                "validators": {
                    "required": true
                }
            },
            {
                "type": "boolean",
                "name": "noSecondName",
                "label": "Без отчества",
                "value": true
            },
            {
                "type": "string",
                "name": "secondName",
                "label": "Отчество",
                "validators": {
                    "required": true
                },
                "effect": {
                    "if": {
                        "operator": "AND",
                        "predicates": [
                            {
                                "attr": "user.noSecondName",
                                "operator": "EQUAL",
                                "value": true
                            }
                        ]
                    },
                    "then": {
                        "disabled": true
                    },
                    "else": {
                        "disabled": false
                    }
                }
            },
            {
                "type": "button",
                "display": "flat",
                "icon": "save",
                "color": "primary",
                "text": "Submit",
                "click": {
                    "call": "MyForm1.validate"
                }
            }
        ]
    }
]
```

```
}
```

Вычисление значения в форме

В данном примере вычисляется значение для поля **Z** через умножение значения в поле **X** на значение в поле **Y**.
Вычисление производится только в том случае, если значения в полях **X** и **Y** больше 0.

```
{
    "data$": {
        "calc": null
    },
    "components": [
        {
            "type": "form",
            "data$": "calc",
            "cols": 3,
            "fields": [
                {
                    "type": "decimal",
                    "name": "x",
                    "label": "X"
                },
                {
                    "type": "decimal",
                    "name": "y",
                    "label": "Y",
                    "prefix": "*"
                },
                {
                    "type": "decimal",
                    "name": "z",
                    "label": "Z",
                    "prefix": "=",
                    "precision": 6,
                    "readonly": true,
                    "effect": {
                        "if": {
                            "operator": "AND",
                            "predicates": [
                                {
                                    "attr": "calc.x",
                                    "operator": "GREATER_THAN",
                                    "value": 0
                                },
                                {
                                    "attr": "calc.y",
                                    "operator": "GREATER_THAN",
                                    "value": 0
                                }
                            ]
                        },
                        "then": {
                            "calculate": "${calc.x} * ${calc.y}"
                        },
                        "else": {
                            "evaluate": null
                        }
                    }
                }
            ]
        }
    ]
}
```

Связанные `lookup`

В данном примере заданы три связанные поля.
Модель нельзя выбрать, если не указана **Марка**.
Модификацию нельзя выбрать, если не указана **Модель**.
При этом, в зависимые поля устанавливается соответствующий фильтр, по выбранной **Марке** или **Модели**.

```
{
    "data$": {
        "car": null
    },
    "components": [
        {
            "type": "form",
            "data$": "car",
            "cols": 3,
            "fields": [
                {
                    "type": "lookup",
                    "name": "brandId",
                    "label": "Марка",
                    "span": 3,
                    "uri": "/sdi/api/sdi-sandbox",
                    "entity": "VehicleBrand",
                    "grid": "main",
                    "dataLoader": {
                        "method": "GET",
                        "uri": "/data/crud/vehicle-brand/{__object_id}"
                    }
                },
                {
                    "type": "lookup",
                    "name": "model",
                    "label": "Модель",
                    "span": 3,
                    "uri": "/sdi/api/sdi-sandbox",
                    "entity": "VehicleModel",
                    "grid": "main",
                    "dataLoader": {
                        "method": "GET",
                        "uri": "/data/crud/vehicle-model/{__object_id}"
                    },
                    "effect": {
                        "if": {
                            "operator": "AND",
                            "predicates": [
                                {
                                    "attr": "car.brandId",
                                    "operator": "IS_NOT_NULL"
                                }
                            ]
                        },
                        "then": {
                            "disabled": false,
                            "filterBy": {
                                "operator": "AND",
                                "predicates": [
                                    {
                                        "attr": "brandId.id",
                                        "operator": "EQUAL",
                                        "value": ["${car.brandId.id}"]
                                    }
```

```
                            ]
                        },
                        "evaluate": null
                    },
                    "else": {
                        "disabled": true,
                        "evaluate": null
                    }
                }
            },
            {
                "type": "lookup",
                "name": "modification",
                "label": "Модификация",
                "span": 3,
                "uri": "/sdi/api/sdi-sandbox",
                "entity": "VehicleModification",
                "grid": "main",
                "dataLoader": {
                    "method": "GET",
                    "uri": "/data/crud/vehicle-modification/{__object_id}"
                },
                "effect": {
                    "if": {
                        "operator": "AND",
                        "predicates": [
                            {
                                "attr": "car.model",
                                "operator": "IS_NOT_NULL"
                            }
                        ]
                    },
                    "then": {
                        "disabled": false,
                        "filterBy": {
                            "operator": "AND",
                            "predicates": [
                                {
                                    "attr": "model",
                                    "operator": "EQUAL",
                                    "value": ["${car.model.id}"]
                                }
                            ]
                        },
                        "evaluate": null
                    },
                    "else": {
                        "disabled": true,
                        "evaluate": null
                    }
                }
            }
        ]
    }
]
}
```

header

Описание

Данный компонент используется для отображения заголовка на странице.

Свойства

Наименование	Тип	Обязательность	Описание	Пример
type	string	Да	Тип компонента	`"type": "header"`
caption	string	Да	Заголовок	`"caption": "Заголовок страницы"`

Поддерживаемые реакции эффектов

Реакция	Тип	Описание
hidden	boolean	Скрыть панель (если `true`)

Примеры

Страница с заголовком и формой

```
{
    "components": [
        {
            "type": "header",
            "caption": "Карточка пользователя"
        },
        {
            "type": "form",
            "fields": [
                {
                    "type": "string",
                    "name": "lastName",
                    "label": "Фамилия"
                },
                {
                    "type": "string",
                    "name": "firstName",
                    "label": "Имя"
                },
                {
                    "type": "string",
                    "name": "secondName",
                    "label": "Отчество"
                }
            ]
        }
    ]
}
```

Связь формы и заголовка

```
{
    "data$": {
        "user": null
    },
    "components": [
        {
            "type": "header",
            "caption": "Карточка пользователя: ${user.lastName} ${user.firstName}
${user.secondName}"
        },
        {
            "type": "form",
            "data$": "user",
            "fields": [
```

```
                {
                    "type": "string",
                    "name": "lastName",
                    "label": "Фамилия"
                },
                {
                    "type": "string",
                    "name": "firstName",
                    "label": "Имя"
                },
                {
                    "type": "string",
                    "name": "secondName",
                    "label": "Отчество"
                }
            ]
        }
    ]
}
```

journal

Описание

Данный компонент используется для отображения гибких таблиц.

Свойства

Наименование	Тип	Обязательность	Описание	Пример
`type`	“journal”	Да	Тип компонента	`"type": "journal"`
`uri`	string	Да		`"uri": "MyForm"`
`entity`	string	Да		`"entity": ""`
`grid`	string	Нет		`"grid": ""`
`keepState`	boolean	Нет	Сохранять состояния журнала	`"keepState": true`
`keepMode`	`QueryParams` | `LocalStorage` | `SessionStorage`	Нет	Режим хранения состояний: `QueryParams` - в виде сжатой строки в query-параметре `_sjsk` адресной строки страницы; `LocalStorage` и `SessionStorage` - в соответствующих хранилищах браузера. Если установлен `keepState` и не задан `keepMode`,	`"keepMode": "QueryParams"`

Наименивание	Тип	Обязательность	Описание	Пример
			то принимается значение по умолчанию `SessionStorage`	
`showToolbar`	boolean	Нет	Отображать штатную панель инструментов	`"showToolbar": true`
`showFilterbar`	boolean	Нет	Отображать вспомогательную панель с установленным и фильтрами	`"showFilterbar": true`
`rowclick`	Action	Нет	Описание действия при нажатии на строку	`"rowclick": {}`
`rowdblclick`	Action	Нет	Описание действия при двойном нажатии на строку	`"rowdblclick": { "navigate": "/entity/form" }`
`contextmenu`	Array<Action>	Нет	Список действий (пункты меню) при вызове контекстного меню	`"contextmenu": [ { "text": "Подробнее...", "icon": "search_activity", "click": { "navigate": "/audit/event/table/form", "params": { "id": "{__object_id}" } } } ]`
`waitingProgramFilter`	boolean	Нет	Журнал не загружает данные, пока не установлен программный фильтр	`"waitingProgramFilter": true`
`selection`	`"single"` | `"multiple"` | `"range"`	Нет	Режим выбора записей: `single` - однострочный, `multiple` - многострочный, `range` - область ячеек	`"selection": "range"`
`copyToClipboard`	boole	Нет	Включить	`"copyToCli`

Наименивание	Тип	Обязательность	Описание	Пример
	an		возможность копирования данных таблицы в буфер обмена	`pboard": true`

По умолчанию, во всех журналах устапновлен режим сохранения состояний в SessionStorage {.is-info}

Методы

Наименивание	Описание
`tools`	Показать штатное меню инструментов ГТ
`reload`	Перезагрузить страницу ГТ
`filter`	Показать штатный фильтр ГТ
`copyPage`	Скопировать в буфер обмена данные текущей страницы
`copyPageWithHeaders`	Скопировать в буфер обмена данные текущей страницы с заголовками
`copySelected`	Скопировать в буфер обмена выделенное
`copySelectedWithHeaders`	Скопировать в буфер обмена выделенное с заголовками

Поддерживаемые реакции эффектов

Реакция	Тип	Описание
hidden	boolean	Скрыть панель (если `true`)
userFilter	JournalFilter | null	Установить пользовательский фильтр.
programFilter	JournalFilter | null	Установить программный фильтр.
purge	boolean	Очистить журнал от данных (если `true`)
reload	boolean	Очистить журнал от данных (если `true`)

Примеры

Простой журнала

```
{
    "type": "journal",
    "entity": "EntityName",
    "grid": "main",
    "uri": "/sdi/api/entity"
}
```

Журнал с переходом по двойному клику на строке

```
{
    "type": "journal",
    "entity": "EntityName",
    "grid": "main",
    "uri": "/sdi/api/entity",
    "rowdblclick": {
```

```
        "navigate": "/entity/form/{__object_id}"
    }
}
```

Журнал с контекстным меню

```
{
    "type": "journal",
    "entity": "EntityName",
    "grid": "main",
    "uri": "/sdi/api/entity",
    "contextmenu": [
        {
            "text": "Добавить",
            "icon": "add",
            "click": {
            "navigate": "/entity/form",
                "params": {
                    "action": "create"
                }
            }
        },
        {
            "text": "Редактировать",
            "icon": "edit",
            "click": {
            "navigate": "/entity/form",
                "params": {
                    "action": "update",
                    "id": "{__object_id}"
                }
            }
        },
        {
            "text": "Удалить",
            "icon": "delete",
            "click": {
            "navigate": "/entity/form",
                "params": {
                    "action": "delete",
                    "id": "{__object_id}"
                }
            }
        }
    ]
}
```

Журнал с указанием механизма хранения состояния

```
{
    "type": "journal",
    "entity": "EntityName",
    "grid": "main",
    "uri": "/sdi/api/entity",
    "keepState": true,
    "keepMode": "QueryParams"
}
```

Master / Detail

```
{
    "data$": {
        "brand": null
    },
```

```
    "components": [
        {
            "type": "journal",
            "entity": "VehicleBrand",
            "grid": "main",
            "uri": "/sdi/api/sdi-sandbox",
            "data$": "brand",
            "showToolbar": true
        },
        {
            "type": "journal",
            "entity": "VehicleModel",
            "grid": "main",
            "uri": "/sdi/api/sdi-sandbox",
            "keepState": false,
            "showToolbar": true,
            "showFilterbar": true,
            "waitingProgramFilter": true,
            "effect": [
                {
                    "if": {
                        "operator": "AND",
                        "predicates": [
                            {
                                "attr": "brand.__object_id",
                                "operator": "IS_NOT_NULL"
                            }
                        ]
                    },
                    "then": {
                        "programFilter": {
                            "operator": "AND",
                            "predicates": [
                                {
                                    "attr": "brandId",
                                    "operator": "EQUAL",
                                    "value": [
                                        "${brand.__object_id}"
                                    ]
                                }
                            ]
                        }
                    },
                    "else": {
                        "programFilter": null
                    }
                }
            ]
        }
    ]
}
```

Журнал с копированием в буфер обмена

Для определения того, что мы копируем в буфер обмена, включён режим выделения данных `range` {.is-info}

```
{
    "components": [
        {
            "type": "journal",
            "entity": "VehicleBrand",
            "grid": "main",
```

```
            "uri": "/sdi/api/sdi-sandbox",
            "selection": "range",
            "copyToClipboard": true
        }
    ]
}
```

Вызов методов журнала

```
{
    "components": [
        {
            "type": "toolbar",
            "tools": [
                {
                    "type": "button",
                    "display": "icon",
                    "icon": "refresh",
                    "click": {
                        "call": "MyJournal.reload"
                    }
                },
                {
                    "type": "button",
                    "display": "icon",
                    "icon": "filter_alt",
                    "click": {
                        "call": "MyJournal.filter"
                    }
                },
                {
                    "type": "button",
                    "display": "icon",
                    "icon": "content_paste",
                    "click": {
                        "call": "MyJournal.copyPage"
                    }
                },
                {
                    "type": "button",
                    "display": "icon",
                    "icon": "settings",
                    "click": {
                        "call": "MyJournal.tools"
                    }
                }
            ]
        },
        {
            "id": "MyJournal",
            "type": "journal",
            "entity": "VehicleModification",
            "grid": "main",
            "uri": "/sdi/api/sdi-sandbox",
            "copyToClipboard": true
        }
    ]
}
```

menu

Описание

Данный компонент используется для отображения меню на странице.

Свойства

Наименивание	Тип	Обязательность	Описание
type	string	Да	Тип компонента
items	Array<MenuItems>	Да	Массив элементов меню

MenuItems

Наименивание	Тип	Обязательность	Описание
text	string	Да	Наименование пункта меню
icon	string	Нет	Иконка пункта меню
click	Action	Нет	Действие пункта меню
items	Array<MenuItems>	Нет	Массив дочерних элементов меню

Поддерживаемые реакции эффектов

Реакция	Тип	Описание
hidden	boolean	Скрыть панель (если `true`)

Пример

Пример страницы с заголовком и формой

```
{
    "components": [
        {
            "type": "menu",
            "items": [
                {
                    "text": "Page1",
                    "icon": "add",
                    "items": [
                        {
                            "text": "Page1.1",
                            "icon": "edit",
                            "click": {
                                "navigate": "page2"
                            }
                        },
                        {
                            "text": "Длинное название пункта меню",
                            "icon": "star",
                            "items": [
                                {
                                    "text": "Page1.2.1",
```

```
                    "icon": "edit",
                    "click": {
                        "navigate": "page2"
                    }
                },
                {
                    "text": "Page1.2.2",
                    "click": {
                        "navigate": "page3"
                    }
                },
                {
                    "text": "Page1.2.3",
                    "icon": "search",
                    "items": [
                        {
                            "text": "Page1.2.3.1",
                            "icon": "edit",
                            "click": {
                                "navigate": "page2"
                            }
                        },
                        {
                            "text": "Page1.2.3.2",
                            "icon": "delete",
                            "click": {
                                "navigate": "page3"
                            }
                        }
                    ]
                }
            ]
        },
        {
            "text": "Page1.3",
            "icon": "search",
            "click": {
                "navigate": "page4"
            },
            "items": [
                {
                    "text": "Page1.3.1",
                    "icon": "edit",
                    "click": {
                        "navigate": "page2"
                    }
                }
            ]
        }
    ]
},
{
    "text": "Длинное название страницы",
    "click": {
        "navigate": "page2"
    }
},
{
    "text": "Page3",
    "click": {
        "navigate": "page3"
    }
},
{
```

```
                    "text": "Page4"
                }
            ]
        }
    ]
}
```

spacer

Описание

Данный компонент используется для отображения пустого блока между отображаемыми компонентами.
Он поддерживается компонентами `toolbar` и `actionbar`.
В компоненте `form` поддерживается свойство `span`.

Свойства

Наименование	**Тип**	**Обязательность**	**Описание**
type	string	Да	Тип компонента (`spacer`)
span	number	Нет	Кол-во занимаемых колонок, при колоночном отображении

Поддерживаемые реакции эффектов

Реакция	**Тип**	**Описание**
hidden	boolean	Скрыть панель (если `true`)

Пример

Пример страницы с заголовком и формой

```
{
    "components": [
        {
            "type": "toolbar",
            "tools": [
                {
                    "type": "header",
                    "caption": "Модель ТС"
                },
                {
                    "type": "spacer"
                },
                {
                    "type": "button",
                    "display": "icon",
                    "icon": "more_vert"
                }
            ]
        }
    ]
}
```

toolbar

Описание

Данный компонент используется для отображения панели инструментов на странице.

> При прокрутке, данный элемент прилипает к верхнему краю экрана.
> {.is-info}

Свойства

Наименивание	Тип	Обязательность	Описание
type	string	Да	Тип компонента (`toolbar`)
tools	Array< Button | Spacer | Header >	Да	Массив элементов панели инструментов

Поддерживаемые реакции эффектов

Реакция	Тип	Описание
hidden	boolean	Скрыть панель (если `true`)

Примеры

Панель инструментов для журнала

```
{
    "components": [
        {
            "type": "toolbar",
            "tools": [
                {
                    "type": "header",
                    "caption": "Модель ТС"
                },
                {
                    "type": "spacer"
                },
                {
                    "type": "button",
                    "display": "icon",
                    "icon": "refresh",
                    "click": {
                        "call": "journal.reload"
                    }
                },
                {
                    "type": "button",
                    "display": "icon",
                    "icon": "filter_alt",
                    "click": {
                        "call": "journal.filter"
                    }
```

```
                },
                {
                    "type": "button",
                    "display": "icon",
                    "icon": "more_vert",
                    "click": {
                        "context": "EVENT",
                        "call": "journal.tools"
                    }
                }
            ]
        },
        {
            "id": "journal",
            "type": "journal",
            "entity": "VehicleModel",
            "grid": "main",
            "uri": "/sdi/api/sdi-sandbox"
        }
    ]
}
```

Действия “Action”

Описание

Данный инструкция описывает действие.

Общие параметры

Наименивание	Тип	Обязательность	Описание
context	string	Нет	Указатель на контекст
chain	Action	Нет	Цепочка действий

Указатель на контекст

Параметр `context` принимает следующие параметры

Параметр	Описание
QUERY_PARAMS	В качестве контекста передаётся Query-параметры
“MyForm”	ID компонента, который предоставляет данные, непример `form`
“${MyData}”	Указатель на блок данных

Примеры указания контекста

```
{
    "data$": {
        "MyData": null
    },
    "components": [
        {
            "type": "container",
            "components": [
                {
                    "type": "form",
                    "id": "MyForm",
                    "data$": "MyData",
                    "cols": 3,
```

```
"api": {
    "read": {
        "method": "GET",
        "uri": "/api/v1/data/service/{id}",
        "params": {
            "id": "{id}",
            "test": true,
            "x": 69
        }
    },
    "write": {
        "method": "POST",
        "uri": "/api/v1/service/entity/save"
    }
},
"fields": [
    {
        "type": "string",
        "name": "field1",
        "label": "Поле 1",
        "span": 3,
        "value": 2
    },
    {
        "type": "string",
        "name": "field2",
        "label": "Поле 2",
        "span": 3,
        "value": 3
    },
    {
        "type": "select",
        "name": "field3",
        "label": "Поле 3",
        "span": 2,
        "options": [
            {
                "id": null,
                "objectName": "-- Выберите значение --"
            },
            {
                "id": 1,
                "objectName": "Значение 1"
            },
            {
                "id": 2,
                "objectName": "Значение 2"
            },
            {
                "id": 3,
                "objectName": "Значение 3"
            },
            {
                "id": 4,
                "objectName": "Значение 4"
            }
        ],
        "validators": {
            "required": true
        }
    },
    {
        "type": "button",
        "text": "Default",
```

```
                "color": "accent",
                "display": "flat",
                "click": {
                    "call": "MyForm.validate",
                    "chain": {
                        "context": "MyForm",
                        "request": {
                            "method": "POST",
                            "uri": "/api/v1/some/service/{field3}"
                        }
                    }
                }
            }
        ]
    },
    {
        "type": "actionbar",
        "buttons": [
            {
                "type": "button",
                "text": "Query Params",
                "click": {
                    "context": "QUERY_PARAMS",
                    "request": {
                        "method": "POST",
                        "uri": "/api/v1/some/service/{id}"
                    }
                }
            },
            {
                "type": "button",
                "text": "Form ID",
                "click": {
                    "context": "MyForm",
                    "request": {
                        "method": "POST",
                        "uri": "/api/v1/some/service/{field1}"
                    }
                }
            },
            {
                "type": "button",
                "text": "Data",
                "click": {
                    "context": "${MyData}",
                    "request": {
                        "method": "POST",
                        "uri": "/api/v1/some/service/{field2}"
                    }
                }
            },
            {
                "type": "button",
                "text": "Validate",
                "click": {
                    "call": "MyForm.validate",
                    "chain": {
                        "context": "${MyData}",
                        "request": {
                            "method": "POST",
                            "uri": "/api/v1/some/service/{field3}"
                        }
                    }
                }
```

```
                        }
                    ]
                }
            ]
        }
    ]
}
```

Действие “Навигация”

Наименивание	Тип	Обязательность	Описание
navigate	string	Да	Адрес перехода
params	Record<string, string	number	boolean>	

Действие “Вызвать метод компонента”

Наименивание	Тип	Обязательность	Описание
call	string	Да	Указатель на метод компонента

Действие “Вызвать backend сервис”

Наименивание	Тип	Обязательность	Описание
request	API	Да	Описание вызова backend сервиса

Действие “Вызвать пользовательскую функцию”

Данное действие работает для встраиваемого в angular-компонент SDI

Наименивание	Тип	Обязательность	Описание
function	Function	Да	Указатель на вызываемую функцию
args	Record<string, string	number	boolean	

Пример

В данном примере для кнопки задано действие, которое вызывает `myFunc` и передаёт туда в качестве аргументов три параметра `api`, `entity` и `id`

```
import { Component } from '@angular/core';
import {
    type SDIButton,
    type SDIForm,
    type SDIPage,
    SdiModule,
} from '@biv/sdi';
import { Observable, of } from 'rxjs';


@Component({
    template: '<sdi-wrapper [instruction]="instruction" />',
    standalone: true,
    imports: [ SdiModule ],
})
export class SdiJournalAdmComponent {

    public readonly instruction: SDIPage = {
        components: [
            {
                type: "form",
```

```
                id: "journal",
                fields: [
                    { type: "string", name: "api", label: "API", value:
'/sdi/api/sdi-sandbox' },
                    { type: "string", name: "entity", label: "Сущность", value:
'VehicleBrand' },
                    { type: "integer", name: "id", label: "ID конфигурации",
value: 11 },
                ]
            } as SDIForm,
            {
                type: "button",
                icon: "settings",
                text: "Click me",
                display: "raised",
                color: "primary",
                click: {
                    context: 'journal',
                    function: this.myFunc,
                    args: {
                        api: '{api}',
                        entity: '{entity}',
                        id: '{id}'
                    }
                } as any
            } as SDIButton
        ]
    }

    public myFunc(api: string, entity: string, id: number): Observable<any> {
        console.log('Clicked with context', api, entity, id);
        return of();
    }
}
```

Действие “Вызвать форму конфигурации гибких таблиц”

Данное действие работает для встраиваемого в angular-компонент SDI

Наименивание	Тип	Обязательность	Описание
custom	entityJournalConfig	Да	
args	{ api: string, entity: string, id: number }	Да	Агрументы для вызова формы конфигурации

Пример

Кнопка, вызывающая форму настройки гибкой таблицы с аргументами `{ api: '/sdi/api/sdi-sandbox', entity: 'VehicleBrand', id: 11 }`

```
import { Component } from '@angular/core';
import {
    type SDIButton,
    type SDIPage,
    SdiModule,
} from '@biv/sdi';
```

```
@Component({
    template: '<sdi-wrapper [instruction]="instruction" />',
    standalone: true,
    imports: [ SdiModule ],
})
export class SdiJournalAdmComponent {

    public readonly instruction: SDIPage = {
        components: [
            {
                type: "button",
                icon: "settings",
                text: "Click me",
                display: "raised",
                color: "primary",
                click: {
                    custom: 'entityJournalConfig',
                    args: {
                        api: '/sdi/api/sdi-sandbox',
                        entity: 'VehicleBrand',
                        id: 11
                    }
                } as any
            } as SDIButton
        ]
    }
}
```

Примеры

Компонент `toolbar` с переходами на страницы

```
{
    "components": [
        {
            "type": "toolbar",
            "tools": [
                {
                    "type": "button",
                    "text": "Page 1",
                    "click": {
                        "navigate": "/page1"
                    }
                },
                {
                    "type": "button",
                    "text": "Page 2",
                    "click": {
                        "navigate": "/page2"
                    }
                },
                {
                    "type": "button",
                    "text": "Page 3",
                    "click": {
                        "navigate": "/page3"
                    }
                }
            ]
        }
    ]
```

```
}
```

Компонент `toolbar` с кнопкой, вызывающей метод `reload` в журнале

```
{
    "components": [
        {
            "type": "toolbar",
            "tools": [
                {
                    "type": "button",
                    "display": "icon",
                    "icon": "refresh",
                    "click": {
                        "call": "journal.reload"
                    }
                }
            ]
        },
        {
            "id": "journal",
            "type": "journal",
            "entity": "VehicleModel",
            "grid": "main",
            "uri": "/sdi/api/sdi-sandbox"
        }
    ]
}
```

Вызов backend сервиса

В данном примере для компонента `journal` описано контекстное меню “**Удалить запись**”. Данный пункт меню вызывает backend сервис и передаёт туда ID записи.
Если сервер возвращает ответ с кодом 200, то отрабатывает цепочка событиый и вызывается метод `reload` журнала `MyJournal`.

```
{
    "components": [
        {
            "id": "MyJournal",
            "type": "journal",
            "entity": "VehicleBrand",
            "grid": "main",
            "uri": "/sdi/api/sdi-sandbox",
            "contextmenu": [
                {
                    "text": "Удалить запись",
                    "icon": "delete",
                    "click": {
                        "request": {
                            "method": "DELETE",
                            "uri": "/vehicle-brand/{__object_id}"
                        },
                        "chain": {
                            "call": "MyJournal.reload"
                        }
                    }
                }
            ]
        }
    ]
}
```

API

Описание

Данный инструкция описывает вызов backend сервиса.

Параметры

Наименование	Тип	Обязательность	Описание
uri	string	Да	URI вызываемого сервиса
method	string	Нет	HTTP-метод. Если не задан, то вызывается как `GET`
params	Record<string, string | number | boolean>	Нет	Query-параметры запроса
request	string	Нет	Указатель на тело запроса
response	string	Нет	Указатель на применение тела ответа
download	boolean	Нет	Вызвать интерфейс скачивания файла (который пришёл в ответе)

Примеры

Заполнение формы данными из указанного источника

```
{
    "components": [
        {
            "type": "form",
            "onInit": {
                "call": "self.read"
            },
            "api": {
                "read": {
                    "method": "GET",
                    "uri": "/api/v1/data/service1/{id}",
                    "response": "self"
                }
            },
            "fields": [
                {
                    "type": "integer",
                    "name": "id",
                    "hidden": true
                },
```

```
                {
                    "type": "string",
                    "name": "field1",
                    "label": "Поле типа \"string\""
                }
            ]
        }
    ]
}
```

В данном примере вызывается метод `read`, описанный в секции `api`.

> Значение `self` ссылается на компонент, в котором описано действие, т.е. на компонент формы. {.is-info}
>
> Значение `self`, указанное в блоке `api -> read -> response`, говорит интерпретатору, взять тело ответа и поместить его в форму. {.is-info}
>
> Параметр `{id}` в атрибуте `api -> read -> uri` подставляется из query-параметров страницы. {.is-info}

Сохранение формы

```
{
    "data$": {
        "formData": null
    },
    "components": [
        {
            "type": "form",
            "id": "MyForm",
            "data$": "formData",
            "api": {
                "save": {
                    "method": "POST",
                    "uri": "/api/v1/service/entity",
                    "params": {
                        "id": "{id}"
                    },
                    "request": "self"
                }
            },
            "cols": 2,
            "fields": [
                {
                    "type": "integer",
                    "name": "id",
                    "value": 13,
                    "hidden": true
                },
                {
                    "type": "string",
                    "name": "field1",
                    "label": "Поле типа \"string\"",
                    "span": 2,
                    "validators": {
                        "required": true
                    }
                }
            ]
        },
        {
```

```
            "type": "actionbar",
            "buttons": [
                {
                    "type": "button",
                    "text": "Сохранить",
                    "click": {
                        "context": "MyForm",
                        "call": "MyForm.save"
                    }
                }
            ]
        }
    ]
}
```

В данном примере в секция `api` задан блок `save`, в котором описан вызов back-end сервиса. Метод `save` вызывается нажатием кнопки **Сохранить**.

> Значение `self`, указанное в блоке `api -> save -> request`, говорит интерпретатору, взять значения формы и передать его в теле запроса. {.is-info}

> Значение `MyForm`, указанное в блоке `click -> context, для конпки **Сохранить**, говорит интерпретатору, использовать значения формы в качестве контекста исполнения вызова (в запрос подставляется query-параметр?id=13`).
> {.is-info}

Данные

Описание

Блок `data$` используется для объявления блока данных с которым взаимодействуют другие компоненты текущей страницы.
Данные блок необходимо объявить в корне инструкции, что бы интерпритатор создал Observable объект при старте.

```
{
    "data$": {
        "myData": null
    }
}
```

Простая связь с блоком данных

Далее на этот блок можно ссылаться, например писать в него данные, вводимые в форму

```
{
    "data$": {
        "myData": null
    },
    "components": [
        {
            "type": "form",
            "data$": "myData",
            "fields": [
                {
                    "type": "string",
                    "name": "lastName",
                    "label": "Фамилия",
                    "validators": {
                        "required": true
```

```
                    }
                }
            ]
        }
    ]
}
```

В компоненте `form`, в параметре `data$`, указано имя блока данных `myData`
{.is-info}

Чтение / Запись

В указателе на блок данных можно описать тип взаимодействия: `read` или `write`.
По умолчанию установлена запись в блок.

```
{
    "data$": {
        "myData": null
    },
    "components": [
        {
            "type": "form",
            "data$": {
                "name": "myData",
                "write": true
            },
            "fields": [
                {
                    "type": "string",
                    "name": "lastName",
                    "label": "Фамилия",
                    "validators": {
                        "required": true
                    }
                }
            ]
        },
        {
            "type": "form",
            "data$": {
                "name": "myData",
                "read": true
            },
            "fields": [
                {
                    "type": "string",
                    "name": "lastName",
                    "label": "Фамилия (чтение)",
                    "readonly": true
                }
            ]
        }
    ]
}
```

На данный момент, не желательно указывать чтение и запись одновременно
{.is-warning}

Данные и Эффекты

Блок данных используется в эффектах.

```
{
    "data$": {
        "calc": null
    },
    "components": [
        {
            "type": "form",
            "data$": "calc",
            "cols": 3,
            "fields": [
                {
                    "type": "decimal",
                    "name": "x",
                    "label": "X"
                },
                {
                    "type": "decimal",
                    "name": "y",
                    "label": "Y",
                    "prefix": "*"
                },
                {
                    "type": "decimal",
                    "name": "z",
                    "label": "Z",
                    "prefix": "=",
                    "precision": 6,
                    "readonly": true,
                    "effect": {
                        "if": {
                            "operator": "AND",
                            "predicates": [
                                {
                                    "attr": "calc.x",
                                    "operator": "GREATER_THAN",
                                    "value": 0
                                },
                                {
                                    "attr": "calc.y",
                                    "operator": "GREATER_THAN",
                                    "value": 0
                                }
                            ]
                        },
                        "then": {
                            "calculate": "${calc.x} * ${calc.y}"
                        },
                        "else": {
                            "evaluate": null
                        }
                    }
                }
            ]
        }
    ]
}
```

В данном примере вычисляется значение для поля **Z** через умножение значения в поле **X** на значение в поле **Y**.
Вычисление производится только в том случае, если значения в полях **X** и **Y** больше 0.
Форма пишет в блок данных `calc`, на который ссылаются проверки и вычисления в блоке `effect`.

Предустановленные данные

Объявляя блок данных можно указать и сами данные.

```
{
    "data$": {
        "user": {
            "lastName": "Иванов",
            "firstName": "Иван",
            "secondName": "Иванович"
        }
    },
    "components": [
        {
            "type": "content",
            "html": "<h1>Привет, ${user.lastName} ${user.firstName} $
{user.secondName}!</h1>"
        }
    ]
}
```

Примеры

Данные и заголовок

```
{
    "data$": {
        "user": null
    },
    "components": [
        {
            "type": "header",
            "caption": "Карточка пользователя: ${user.lastName} ${user.firstName}
${user.secondName}"
        },
        {
            "type": "form",
            "data$": "user",
            "fields": [
                {
                    "type": "string",
                    "name": "lastName",
                    "label": "Фамилия"
                },
                {
                    "type": "string",
                    "name": "firstName",
                    "label": "Имя"
                },
                {
                    "type": "string",
                    "name": "secondName",
                    "label": "Отчество"
                }
            ]
        }
    ]
}
```

Данные и форма

```
{
    "data$": {
```

```
        "user": null
    },
    "components": [
        {
            "type": "form",
            "data$": "user",
            "fields": [
                {
                    "type": "string",
                    "name": "lastName",
                    "label": "Фамилия"
                },
                {
                    "type": "string",
                    "name": "firstName",
                    "label": "Имя"
                },
                {
                    "type": "string",
                    "name": "secondName",
                    "label": "Отчество"
                }
            ]
        },
        {
            "type": "content",
            "html": "<h1>Привет, ${user.lastName} ${user.firstName} $
{user.secondName}!</h1>"
        }
    ]
}
```

История версий

17.0.9

1. Реализовано поле формы типа autocomplete

17.0.8

1. Обновление библиотеки ГТ до **17.5.5**
2. Компоненту журнал добавлены методы: copyPageWithHeaders, copySelected, copySelectedWithHeaders
3. Свойство copyHeaderToClipboard удалено из журнала

17.0.7

1. Обновление библиотеки ГТ до **17.5.4**
2. Обновление документации для ГТ
3. Добавление режима выбора данных `"selection": "range"`
4. Добавление свойств `copyToClipboard`, `copyHeaderToClipboard` для управления буфером обмена

17.0.6

1. Замена библиотеки ГТ @biv/smart-journal на @biv/entity-journal

17.0.5

1. Поправлен вызов кастомного действия в журнале для одиночного и двойного нажатия

на записи

17.0.4

1. Исправлена ошибка передачи контекста в контекстном меню

17.0.3

1. Реализован вызов пользовательских функций `function` для Action
2. Реализован вызов формы конфигурации гибких таблиц `entityJournalConfig` через Action

17.0.2

1. В `sdi-wrapper` добавлено свойство `instruction` (2way binding)
2. В `sdi-wrapper` добавлено свойство `observeRouting`, признак отслеживания роутинга приложения (по умолчанию: `false`)
3. Исправлена ошибка, когда в момент инициализации SDI, не поддгружалась инструкция с бэка по адресу текущему страницы
4. Исправлена ошибка мультизапросов при переключении страниц с разными инстансами SDI

17.0.1

1. Зависимость от библиотеки гибких таблиц “@biv/smart-journal”: “^17.4.0”
2. Возможность указывать тему для журналов через параметер `theme`

17.0.0

- Поехали!