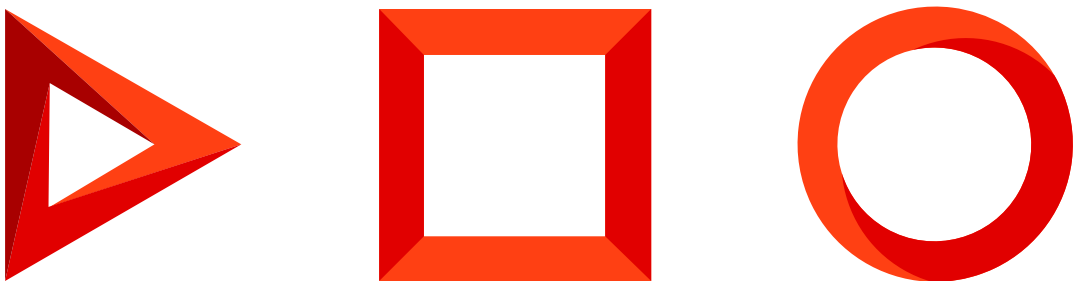
Creatio Academy

Front-end разработка Freedom UI

Front-end архитектура Creatio

Версия 8.0



Эта документация предоставляется с ограничениями на использование и защищена законами об интеллектуальной собственности. За исключением случаев, прямо разрешенных в вашем лицензионном соглашении или разрешенных законом, вы не можете использовать, копировать, воспроизводить, переводить, транслировать, изменять, лицензировать, передавать, распространять, демонстрировать, выполнять, публиковать или отображать любую часть в любой форме или посредством любые значения. Обратный инжиниринг, дизассемблирование или декомпиляция этой документации, если это не требуется по закону для взаимодействия, запрещены.

Информация, содержащаяся в данном документе, может быть изменена без предварительного уведомления и не может гарантировать отсутствие ошибок. Если вы обнаружите какие-либо ошибки, сообщите нам о них в письменной форме.

Содержание

Front-end архитектура Creatio 4

Режим DesignTime 4

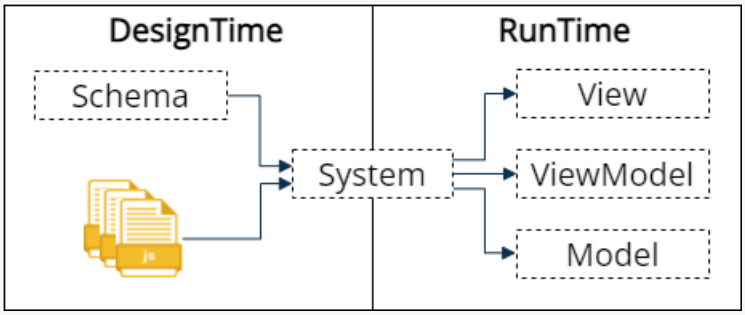
Режим RunTime 4

Front-end архитектура Creatio

Основы

Программная платформа — среда, которая используется для разработки (режим `DesignTime`) и выполнения (режим `RunTime`) приложения.

Схема взаимодействия структурных элементов программной платформы Creatio представлена на рисунке ниже.



Структурные элементы режимов платформы Creatio взаимодействуют через системный слой `System`.

Режим DesignTime

Назначение режима `DesignTime` — разработка, изменение и кастомизация приложения.

Структурные элементы режима `DesignTime` программной платформы Creatio:

- Слой `Schema` — слой метаданных. Содержит набор клиентских схем. Подробнее о клиентских схемах читайте в статье Клиентская схема.
- Предварительно скомпилированный JavaScript-код.

Режим RunTime

Структурные элементы режима `RunTime` программной платформы Creatio:

- Слой `View` — слой визуального представления информации.
- Слой `ViewModel` — слой бизнес-логики взаимодействия слоев `View` и `Model`.
- Слой `Model` — слой данных.

Слой View

`View` — слой, который отвечает за визуальное представление информации. Представлен набором визуальных компонентов режима `DesignTime`.

Для front-end разработки на платформе Creatio Freedom UI появляется понятие декораторов. **Декораторы** — элементы, которые отвечают за регистрацию и расширение функциональности элементов (компонентов, запросов, обработчиков запросов и т. д.) приложения. Использование декораторов позволяет универсиализировать разработку путем использования специальных реестров типов, а также уйти от необходимости подключения функциональности вручную.

Слой `View` реализуют компоненты разных типов. Например, это могут быть компоненты для размещения вложенных компонентов (например, `GridContainerComponent`), компоненты для отображения информации (например, `LabelComponent`), компоненты для взаимодействия с пользователем (например, `InputComponent`) и т. д.

Компоненты могут использоваться в свойстве `viewConfigDiff` клиентской схемы. При использовании компонентов в клиентских схемах компоненты генерируются на основе метаданных схемы, что позволяет кастомизировать визуальное представление с использованием low-code / no-code инструментов Creatio.

Слой `ViewModel`

`ViewModel` — слой, который отвечает за бизнес-логику взаимодействия слоев `View` и `Model`. Представлен типом `ViewModel`, который инкапсулирует в себе логику работы с атрибутами (инициализация данных, привязка к свойствам визуальных компонентов и отслеживание изменений). Creatio не предоставляет возможность создания новых типов `view model`.

Типы компонентов, которые позволяет реализовать слой `ViewModel`:

- **Валидаторы** — функции проверки корректности значения атрибута `ViewModel`. Подробнее читайте в пункте Валидаторы.
- **Конвертеры** — функции модификации значения атрибута `ViewModel`, который привязан к свойству визуального компонента. Подробнее читайте в пункте Конвертеры.
- **Запросы** и **обработчики запросов** — элементы механизма `HandlerChain`, который позволяет описывать бизнес-логику в формате запроса на действие и цепочки обработчиков запроса. Подробнее читайте в пункте Запросы и обработчики запросов.

Каждому типу соответствует декоратор.

Типы атрибутов, которые реализуют слой `ViewModel`:

- Атрибут простого типа (`string`, `number`, `boolean`).
- Атрибут, который содержит вложенные `view model`.
- Ресурсный атрибут (`readonly`).

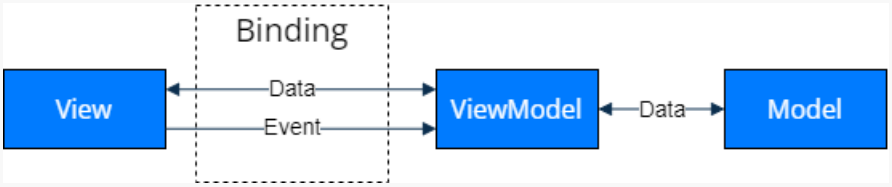
Этапы жизненного цикла, которые поддерживает слой `ViewModel` в Creatio:

- Инициализация экземпляра (`crt.HandleViewModelInitRequest`).
- Изменение значения атрибута (`crt.HandleViewModelAttributeChangeRequest`).
- Уничтожение экземпляра (`crt.HandleViewModelDestroyRequest`). На этом этапе необходимо выполнять только синхронный код, который уничтожает накопленные в процессе работы ресурсы.

Для обеспечения отображения данных в пользовательском интерфейсе приложения и синхронизации

этих данных необходимо установить привязку слоя `View` к `ViewModel`.

Схема привязки `View` к `ViewModel` представлена на рисунке ниже.



Типы привязок, которые предоставляет Creatio:

- Односторонняя привязка к атрибуту.
- Привязка к ресурсному атрибуту.
- Получение `CrtControl` экземпляра.

Пример использования разных типов привязок приведены ниже.

Пример использования привязок

```
schema{
    resources: {
        strings: {
            Title: {
                "en-US": "Example"
            }
        }
    }
    body: define("Example", [], () => {
        viewModelConfig: {
            attributes: {
                FirstName: {},
                Visible: {}
            }
        }
        viewConfigDiff: [{
            name: "Example",
            type: "crt.Input",
            control: "$FirstName", <== CrtControl
            title: "$Resources.Strings.Title", <== Resources
            visible: "$Visible" <== OneWay
        }]
    })
}
```

Механизм привязок можно расширить механизмом макросов, который необходимо использовать,

например, для привязки вложенных свойств объекта к ресурсам `ViewModel`. **Макрос** — элемент, который заменяет часть `view config` на значение ресурсов из `ViewModel`. В отличие от привязки, макрос срабатывает только один раз и в дальнейшем не синхронизируется при изменении `view model`. В Creatio 8.0 Atlas реализован только макрос `#ResourceString#`, в котором реализована работа со строками из ресурсов.

Пример установки значения из ресурсов в свойство `caption` элемента `Header` приведен ниже.

Пример использования макроса `#ResourceString#`

```
viewConfigDiff: /**SCHEMA_VIEW_CONFIG_DIFF*/[
    ...
    {
        "operation": "insert",
        "name": "Header",
        "values": {
            "type": "crt.Label",
            "caption": "#ResourceString(Header)#",
        },
    ...
    },
    ...
]/**SCHEMA_VIEW_CONFIG_DIFF*/
```

Поскольку у пользователей отсутствует возможность создания `ViewModel`, то в Creatio 8.0 Atlas бизнес-логику необходимо описывать в отдельных обработчиках запросов. Обработчики можно объединять в цепочки и определять необходимое время вызова соответствующего обработчика. При создании обработчика можно ограничить область его срабатывания путем добавления в свойство `scopes` имен схем, для которых он должен срабатывать.

Валидаторы

Валидаторы — функции проверки корректности значения атрибута `ViewModel`. Примеры валидаторов: `MaxLengthValidator`, `MinLengthValidator`, `RequiredValidator`.

Валидаторы применяются к атрибутам `ViewModel`, а не к визуальным элементам, но могут получить информацию о статусе валидности через `CrtControl`.

Конвертеры

Конвертеры — функции модификации значения атрибута `ViewModel`, который привязан к свойству визуального компонента. Примеры конвертеров: `crt.invertBooleanValue`, `crt.toBoolean`.

Особенности использования конвертеров:

- В Creatio 8.0 Atlas применяются только в режиме `RunTime`.
- Используются только при установленной привязке, т. е. не работают с константами.

- Работают только в одну сторону. Поэтому недоступны к использованию с `CrtControl`.

Запросы и обработчики запросов

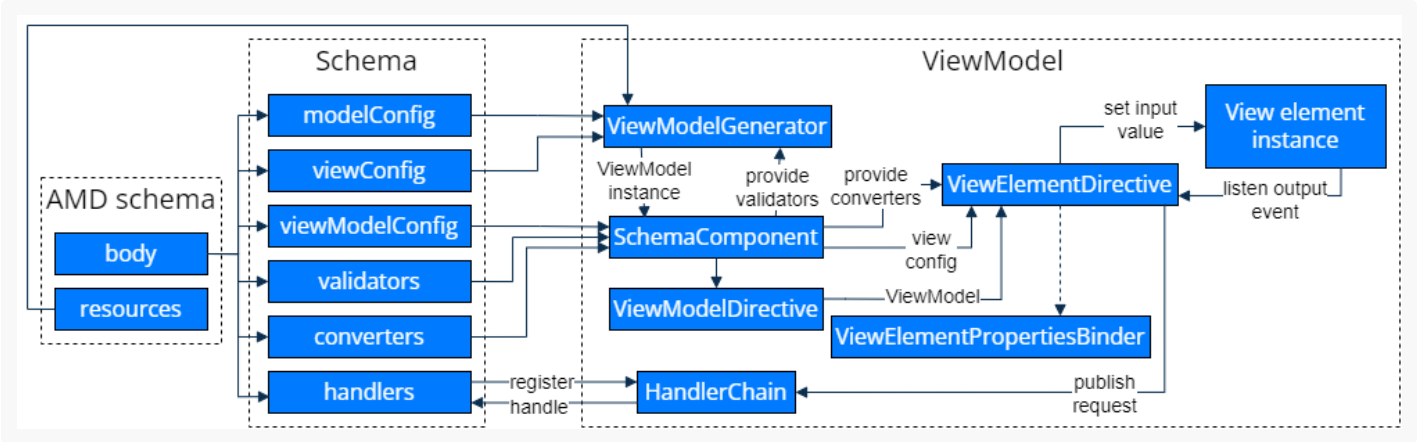
Запросы и **обработчики запросов** — элементы механизма `HandlerChain`, который позволяет описывать бизнес-логику в формате запроса на действие и цепочки обработчиков запроса. Примеры запросов: готовность страницы, загрузка и сохранение данных, запуск бизнес-процесса.

Запросы, кастомизацию которых может выполнять пользователь, приведены в таблице ниже.

Запросы, кастомизацию которых может выполнять пользователь

Тип запроса	Обработчик	Описание
Действия по открытию страниц	`crt.CreateRecordRequest`	Создать запись.
	`crt.UpdateRecordRequest`	Обновить запись.
	`crt.OpenPageRequest`	Открыть страницу.
Действия по работе с данными на странице	`crt.SaveRecordRequest`	Сохранить данные.
	`crt.CancelRecordChangesRequest`	Отменить изменение данных.
Другие действия в дизайнере интерфейсов	`crt.RunBusinessProcessRequest`	Запустить бизнес-процесс.
	`crt.ClosePageRequest`	Закрыть страницу.

Схема работы приложения представлена на рисунке ниже.

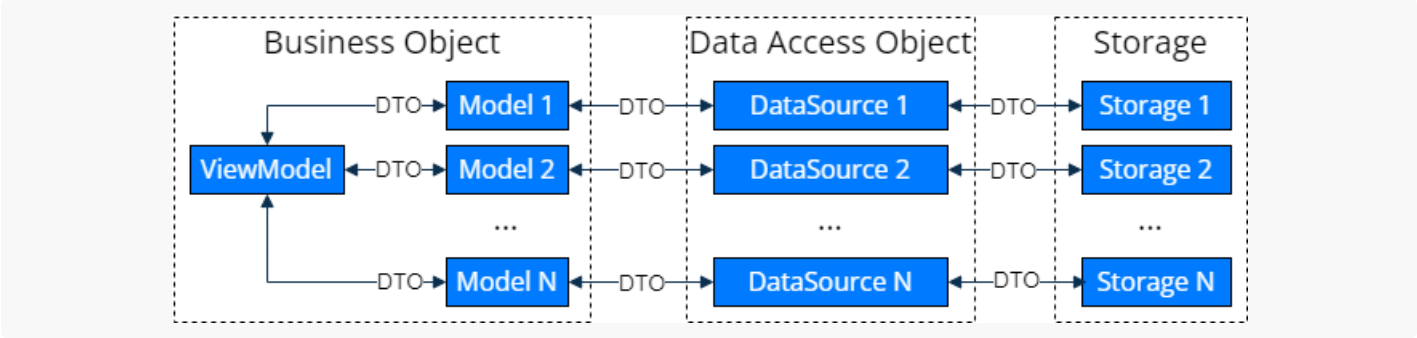


Слой `Model`

`Model` — слой, который отвечает за работу с данными. Позволяет работать с источниками данных, схемой их данных и выполнять операции с данными (загрузка, сохранение, удаление, сортировка и т. д.). В Creatio 8.0 Atlas реализован тип `EntityDataSource` источника данных, который позволяет работать с данными `Entity`-объектов Creatio. Creatio 8.0 Atlas не предоставляет возможность расширения набора источников данных. В дальнейшем эта возможность будет предоставлена. Функциональность слоя `Model` используется слоем `ViewModel` для обеспечения данными слоя `View`.

Для хранения данных в Creatio используется паттерн **Data Access Object** (DAO). Подробнее о паттерне DAO читайте на Википедии.

Схема работы DAO в Creatio представлена на рисунке ниже.



Задачи элементов группы `Data Access Object`:

- Обеспечить выполнение CRUD-операций.
- Предоставить права на операции с данными (создание, редактирование, удаление).
- Предоставить структуру данных (`DataSchema`).

В Creatio 8.0 Atlas реализован `EntityDataSource`, который работает с базой данных приложения.

Инициализация `Model` выполняется с `viewModel` по схеме, которая представлена ниже.