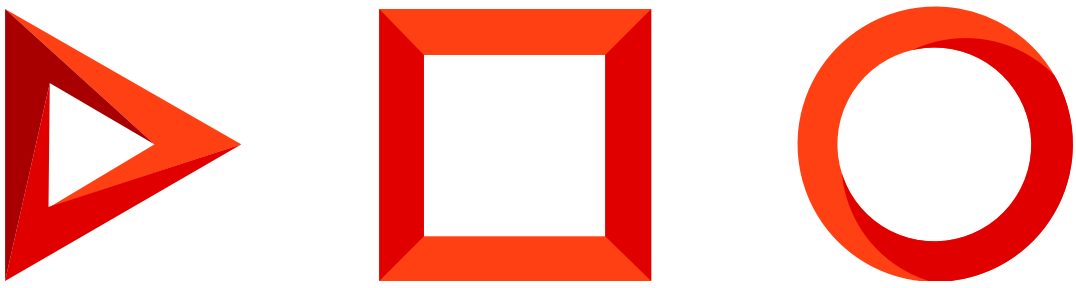
Creatio Academy

Общие принципы работы с пакетами

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

Общие принципы работы с пакетами

Основы

Любой продукт Creatio представляет собой определенный набор пакетов.

Пакет Creatio — это совокупность конфигурационных элементов, которые реализуют блок функциональности. Физически пакет представляет собой папку, содержащую определенный набор вложенных папок и файлов.

Классификация пакетов

- **Простые пакеты**:
  - — предустановленные пакеты. Являются частью приложения и по умолчанию устанавливаются в рабочее пространство. Недоступны для изменения.

    Виды предустановленных пакетов:

    - Пакеты с базовой функциональностью (например, `Base`, `NUI`).
    - Пакеты сторонних разработчиков.
      Устанавливаются из *.zip-архивов с помощью Creatio IDE или с помощью утилиты WorkspaceConsole.
  - — пользовательские пакеты. Созданы другими пользователями системы и заблокированы для изменения в системе контроля версий. Недоступны для изменения.
  - — пользовательские пакеты. Созданы текущим пользователем либо загружены из системы контроля версий. Доступны для изменения.
- **Пакет-проект** — пакет, который позволяет разрабатывать функциональность как обычный C#-проект. Подробнее читайте в статье Пакет-проект.
- **Пакет-сборка** — пакет, исходный и автогенерируемый код которого компилируется в отдельную сборку. Возможность использования пакета-сборки доступна для приложений Creatio версии 7.18.3 и выше. В Creatio IDE пакет-сборка отображается как пользовательский пакет (). Подробнее читайте в статье Пакет-сборка.

Сравнение пакетов Creatio представлено в таблице ниже.

Сравнение пакетов Creatio

Ссылки на другие пакеты	Конфигурационные элементы пакета	Файловый контент	Разработка в Creatio IDE	Путь для компиляции кода
Пакет				
+	+	+	+	`...\Terrasoft.WebApp\Terrasoft.Configuration\Terrasoft.Configuration.dl`
Пакет-проект				
–	–	+	– (используется внешняя IDE)	Для .NET Framework: `...\Terrasoft.WebApp\Terrasoft.Configuration\Pkg\[Имя пакета]\Files\Bin[Имя пакета].dll` Для .NET Core: `...\Terrasoft.WebApp\Terrasoft.Configuration\Pkg\[Имя пакета]\Files\Binnetstandard\[Имя пакета].dll`
Пакет-сборка				
+	+ (с ограничениями)	+	+	Для .NET Framework: `...\Terrasoft.WebApp\Terrasoft.Configuration\Pkg\[Имя пакета]\Files\Bin[Имя пакета].dll` Для .NET Core: `...\Terrasoft.WebApp\Terrasoft.Configuration\Pkg\[Имя пакета]\Files\Binnetstandard\[Имя пакета].dll`

В этой статье рассматриваются простые пакеты.

Для расширения или изменения функциональности необходимо установить пакет с требуемой функциональностью. Разработка дополнительной функциональности и модификация существующей выполняется исключительно в пользовательских пакетах.

Основные пакеты приложения

К основным пакетам приложения можно отнести пакеты, которые обязательно присутствуют во всех продуктах.

Основные пакеты приложения

Название пакета	Описание
`Base`	Базовые схемы основных объектов, разделов системы и связанных с ними схем объектов, страниц, процессов и т. д.
`Platform`	Модули и страницы мастера разделов, дизайнеров реестра и итогов и т. п.
`Managers`	Клиентские модули менеджеров схем.
`NUI`	Функциональность, связанная с пользовательским интерфейсом системы.
`UIv2`	
`DesignerTools`	Схемы дизайнеров и их элементов.
`ProcessDesigner`	Схемы дизайнера процессов.

Пакет `Custom`

В процессе работы мастер разделов или мастер деталей создает схемы, которые необходимо сохранить в пользовательский пакет. В только что установленном приложении нет пакетов, доступных для изменения, а в предустановленные пакеты невозможно внести изменения. Для этого предназначен специальный предустановленный пакет `Custom`. Он позволяет добавлять схемы как вручную, так и с помощью мастеров.

Особенности пакета `Custom`:

- Пакет `Custom` невозможно добавить в систему контроля версий. Поэтому его схемы можно перенести на другую рабочую среду только при помощи функциональности экспорта и импорта пакетов.
- В отличие от других предустановленных пакетов, пакет `Custom` невозможно выгрузить в файловую систему при помощи утилиты WorkspaceConsole.
- В пакете `Custom` установлены зависимости от всех предустановленных пакетов приложения. При создании или установке пользовательского пакета в пакет `Custom` автоматически добавляется зависимость от пользовательского пакета. Таким образом пакет `Custom` всегда должен быть последним в иерархии пакетов.
- В зависимости пользовательских пакетов невозможно добавить пакет `Custom`.

Рекомендуемые **варианты использования** пакета `Custom`:

- Не предполагается перенос изменений в другую рабочую среду.
  В процессе работы мастер разделов или мастер деталей не только создает различные схемы, но и

привязывает данные к текущему пакету. Для пакета `Custom` не предусмотрено использование стандартного механизм импорта пакетов. Поэтому если текущим пакетом является пакет `Custom`, то перенести привязанные данные в другой пользовательский пакет можно только с помощью запросов к базе данных. Мы настоятельно не рекомендуем использовать этот способ, поскольку изменения могут повлиять на структуру базы данных, что приведет к неработоспособности приложения.

При значительной доработке пользовательской функциональности необходимо создать пользовательский пакет с использованием системы контроля версий.

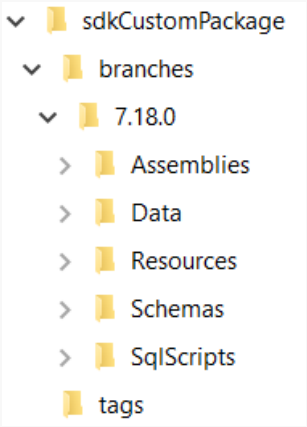
- Изменения выполняются при помощи мастеров или вручную, при этом объем изменений небольшой.
- Нет необходимости использовать систему контроля версий.

Пользовательский пакет

Чтобы выполнять разработку в пользовательском пакете, необходимо в системной настройке [*Текущий пакет*] (код `CurrentPackageId`) указать имя пользовательского пакета.

Структура пакета

При фиксации пакета в системе контроля версий в хранилище пакета создается папка с именем пакета.



Структура папки с именем пакета:

- Папка `branches`.
  Назначение — хранение версий текущего пакета. Версия пакета — отдельная вложенная папка, имя которой совпадает с номером версии пакета в системе (например, 7.18.0).
- Папка `tags`.
  Назначение — хранение меток. **Метки** в системе контроля версий — это "снимок" проекта в определенный момент времени, статическая копия файлов, необходимая для фиксации этапа разработки.

В Creatio по умолчанию включен режим работы с SVN. Для настройки работы с SVN необходимо изменить значение атрибута `connectionString` настройки `defPackagesWorkingCopyPath` конфигурационного файла `ConnectionStrings.config`. Эта настройка содержит путь к каталогу на диске, в котором размещаются рабочие копии.

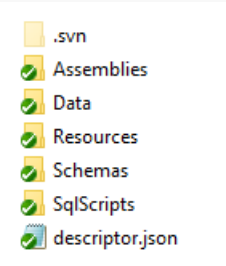
`ConnectionStrings.config`

```
<add name="defPackagesWorkingCopyPath" connectionString="TEMP\APPLICATION\WORKSPACE\TerrasoftPac
```

В режиме работе с SVN приложение Creatio использует собственную рабочую копию каждого пользовательского пакета, для которого подключена версионность. Содержимое рабочей копии представляет собой пользовательские пакеты в виде набора папок и файлов. SVN-клиент, встроенный в Creatio, синхронизирует это содержимое с SVN-репозиторием. В качестве значения настройки `defPackagesWorkingCopyPath` рекомендуется установить путь на фиксированный каталог, поскольку временный каталог, установленный по умолчанию, может быть очищен операционной системой. Для приложения Creatio на платформе .NET Framework в качестве каталога для хранения рабочих копий запрещено указывать `...\Terrasoft.WebApp\Terrasoft.Configuration\Pkg`.

Структура папки пакета в файловой системе:

- Папка `Schemas` — содержит схемы пакета.
- Папка `Assemblies` — содержит внешние сборки, привязанные к пакету.
- Папка `Data` — содержит данные, привязанные к пакету.
- Папка `SqlScripts` — содержит SQL-сценарии, привязанные к пакету.
- Папка `Resources` — содержит локализованные ресурсы пакета.
- Папка `Files` — содержит файловый контент пакета.
- Файл `descriptor.json` — хранит метаданные пакета в формате JSON. К метаданным пакета относятся идентификатор, наименование, версия, зависимости и т. д.



Зависимости и иерархия пакетов

Разработка приложения Creatio базируется на основных принципах проектирования программного обеспечения, в частности, **принципа отсутствия повторений (DRY)**.

В архитектуре Creatio этот принцип реализован с помощью **зависимостей пакетов**. Каждый пакет содержит определенную функциональность приложения, которая не должна повторяться в других пакетах. Чтобы такую функциональность можно было использовать в другом пакете, необходимо пакет, содержащий эту функциональность, добавить в зависимости пакета, в котором она будет использована.

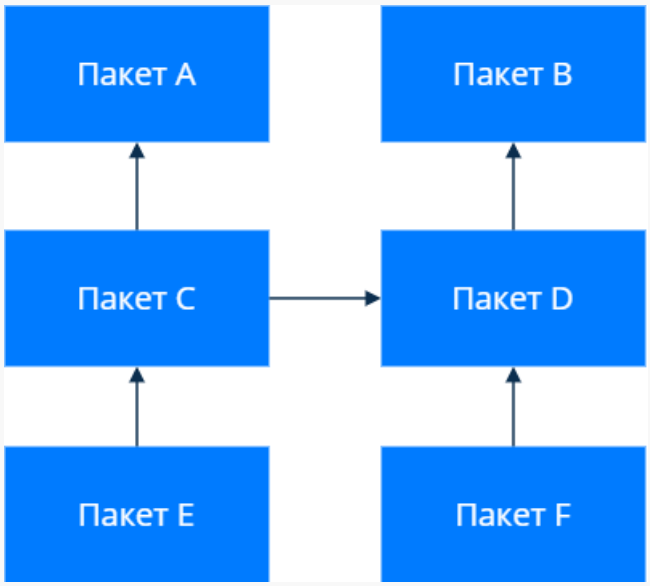
Виды зависимостей:

- Чтобы текущий пакет наследовал всю **функциональность приложения**, в качестве родительского

пакета необходимо выбрать пакет, который в иерархии находится следующим после пакета `Custom`.

- Чтобы текущий пакет наследовал **функциональность пакета**, в качестве родительского пакета необходимо выбрать пакет, функциональность которого необходимо наследовать.

Пакет может иметь несколько зависимостей. Например, в пакете C установлены зависимости от пакетов A и D. Таким образом, вся функциональность пакетов A и D доступна в пакете C.



Зависимости пакетов формируют **иерархические цепочки**. Это означает, что в пакете доступна не только функциональность дочернего пакета, но и функциональность всех пакетов, для которых дочерний пакет является родительским. Ближайшей аналогией иерархии пакетов является иерархия наследования классов в объектно-ориентированном программировании. Так, например, в пакете E доступна функциональность не только пакета C, от которого он зависит, но и функциональность пакетов A, B и D. A в пакете F доступна функциональность пакетов B и D.

Иерархия пакетов приложения

Иерархия и зависимости пакетов отображены на **диаграмме зависимостей пакетов**. Чтобы открыть диаграмму:

1. Перейдите в раздел [*Конфигурация*] ([*Configuration*]).
2. В выпадающем списке [*Действия*] ([*Actions*]) панели инструментов в группе [*Пакеты*] ([*Packages*]) выберите [*Диаграмма зависимостей пакетов*] ([*Package dependencies diagram*]).

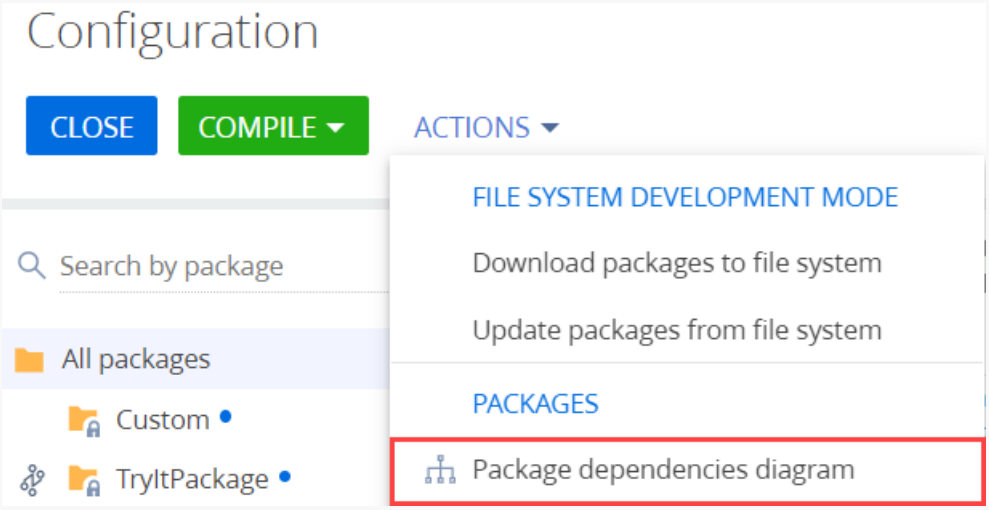
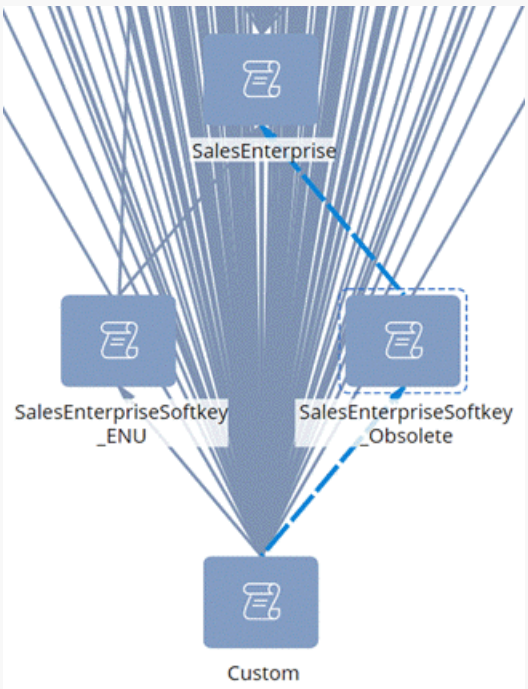


Диаграмма зависимостей будет открыта в новой вкладке.

Если кликнуть по узловому элементу диаграммы с именем пакета, то в виде анимированных стрелок отобразятся связи с другими пакетами. Например, в продукте SalesEnterpise пакет `SalesEnterpriseSoftkey_Obsolete` зависит только от пакета `SalesEnterpise` и всех его родительских пакетов. Также пакет `SalesEnterpriseSoftkey_Obsolete` является родительским для пакета `Custom`.



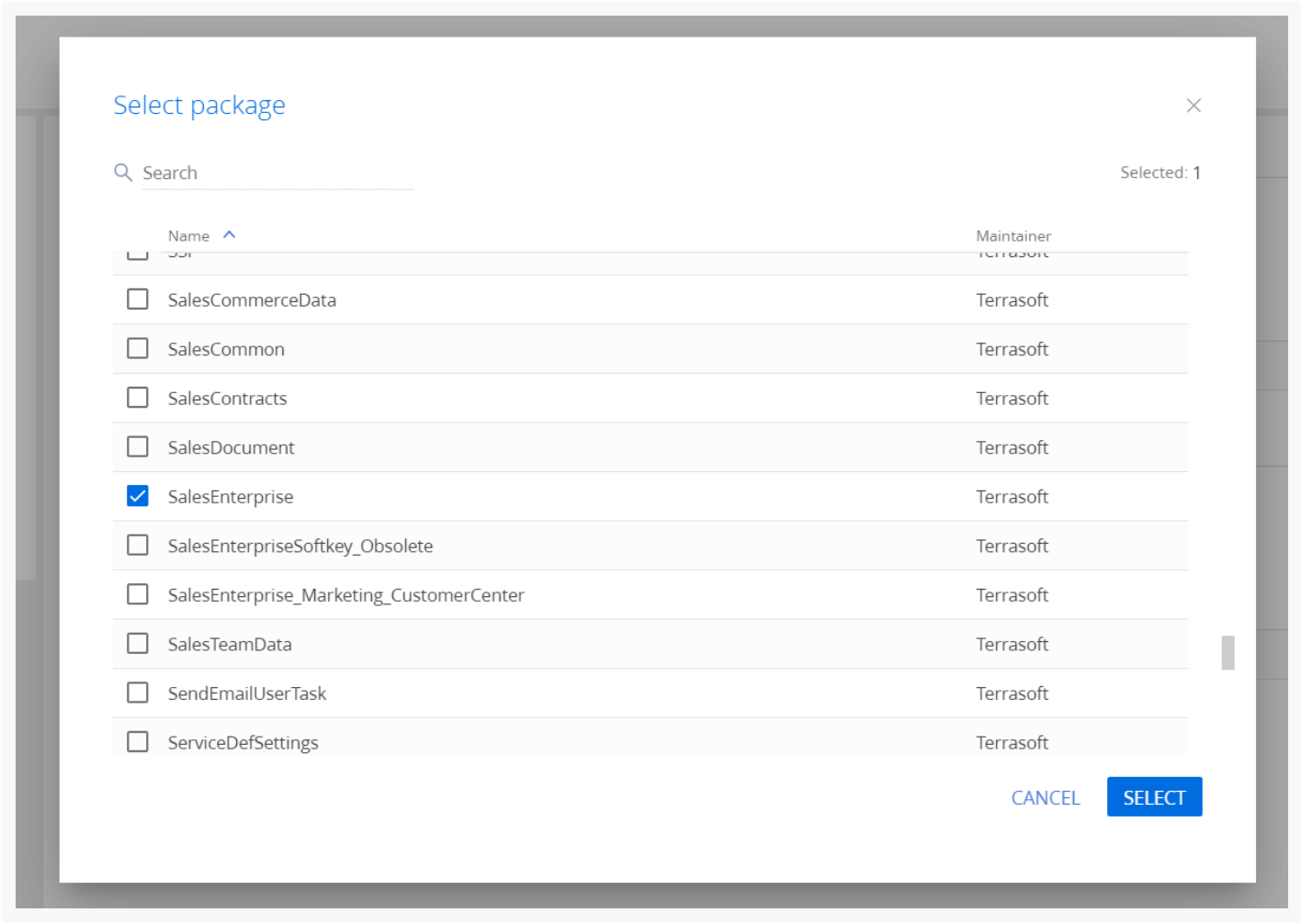
Добавление зависимостей пакета

Зависимости можно добавить в пользовательский пакет при создании пакета или уже после него.

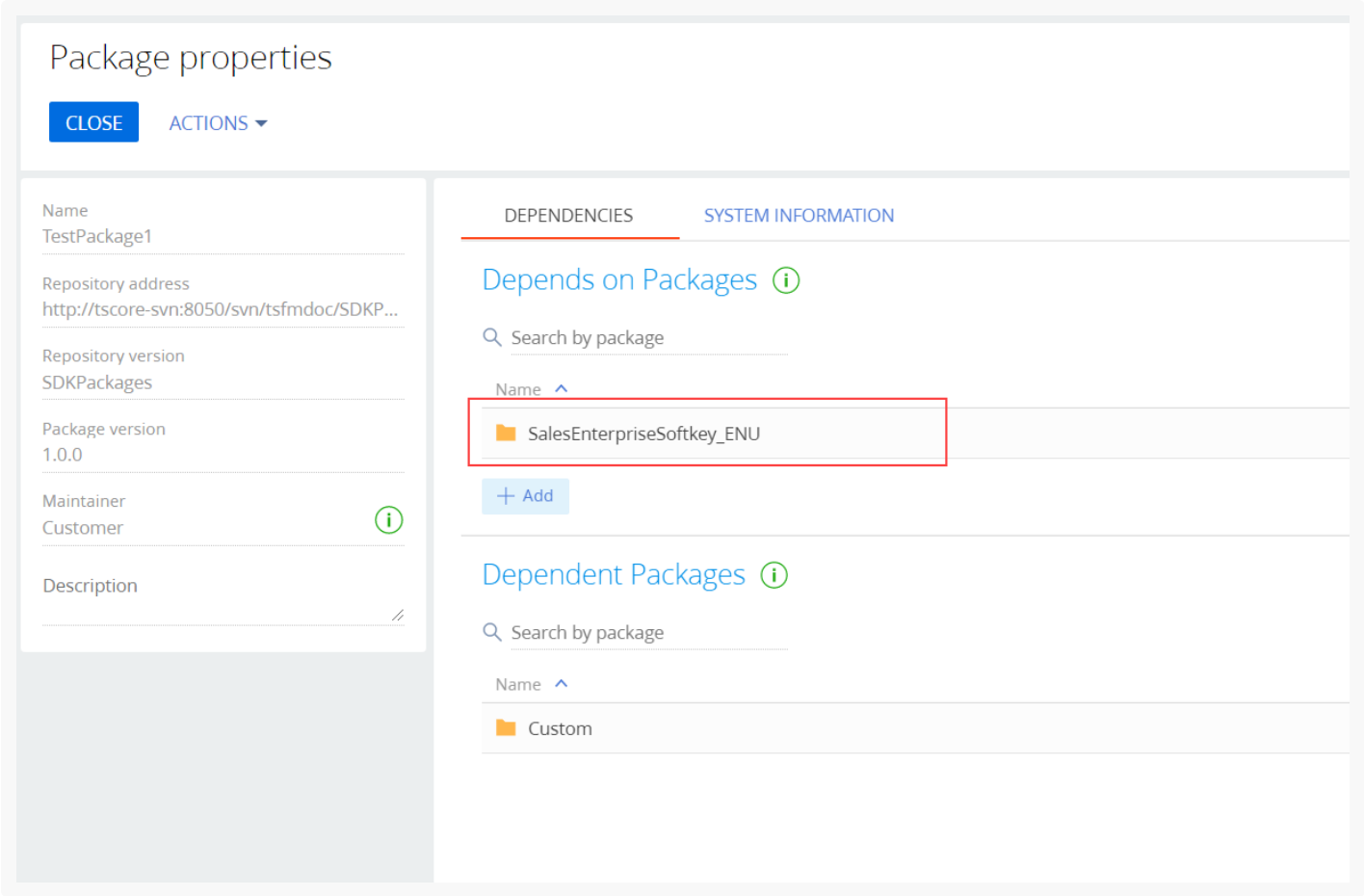
Чтобы **добавить зависимости**:

1. Перейдите на страницу пакета.

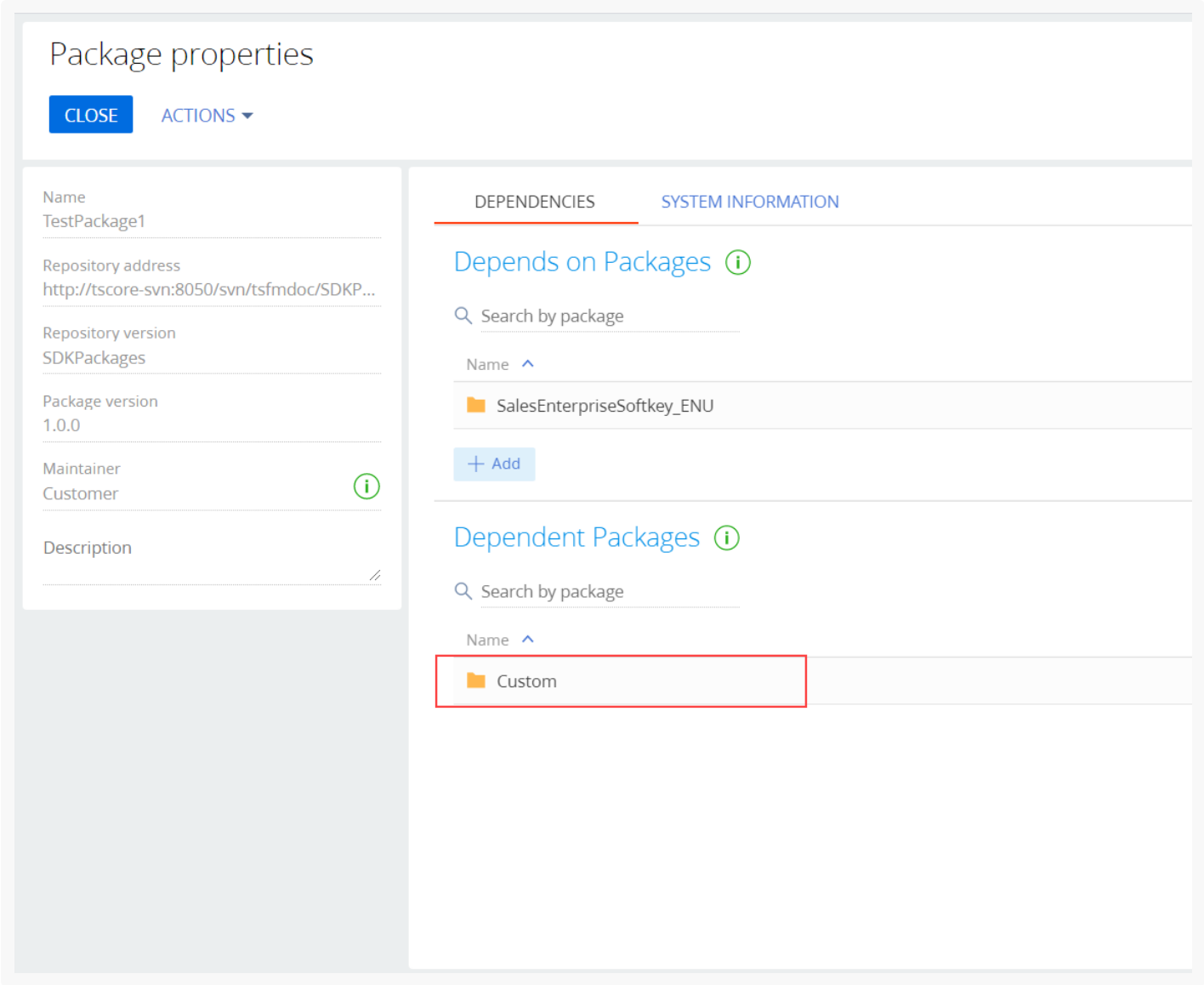
2. На вкладке [*Зависимости*] ([*Dependencies*]) на детали [*Зависит от пакетов*] ([*Depends on packages*]) нажмите кнопку [*Добавить*] ([*Add*]).
3. В появившемся окне справочника пакетов выберите необходимый пакет и нажмите кнопку [*Выбрать*] ([*Select*]).



После этого выбранный пакет будет отображен в списке зависимостей текущего пакета, а при добавлении новой зависимости он будет скрыт из справочника пакетов.



После создания пакет автоматически добавляется в зависимости предустановленного пакета `Custom`.



Список зависимостей в метаданных пакета

Список зависимостей хранится в **метаданных пакета**, которые можно посмотреть в свойстве `DependsOn` объекта, определенного в файле `descriptor.json`.

Свойство `DependsOn` — массив объектов, в которых указывается имя пакета, его версия и уникальный идентификатор, по которому можно определить пакет в базе данных приложения. Файл `descriptor.json` создается приложением для каждой версии пакета.

Пример файла `descriptor.json`

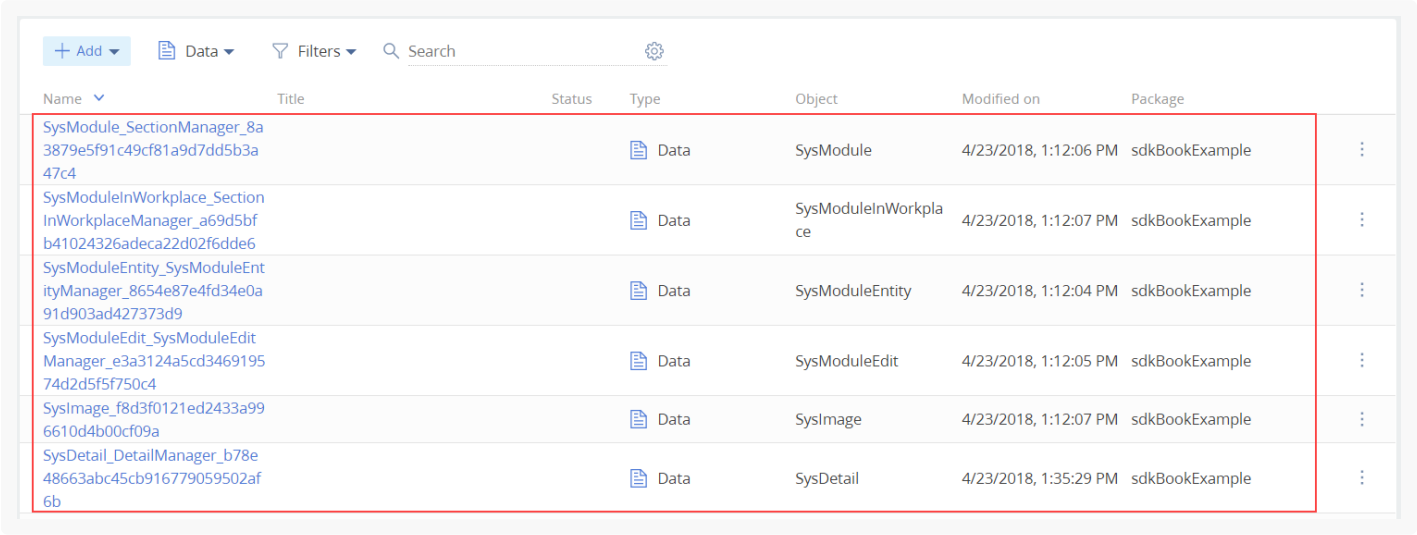
```
{
    "Descriptor": {
        "UId": "51b3ed42-678c-4da3-bd16-8596b95c0546",
        "PackageVersion": "7.18.0",
        "Name": "UsrDependentPackage",
        "ModifiedOnUtc": "\/Date(1522653150000)\/",
```

```
        "Maintainer": "Customer",
        "DependsOn": [
        {
            "UId": "e14dcfb1-e53c-4439-a876-af7f97083ed9",
            "PackageVersion": "7.18.0",
            "Name": "SalesEnterprise"
        }
        ]
    }
}
```

Привязка данных к пакету

При переносе изменений между рабочими средами часто возникает необходимость вместе с разработанной функциональностью предоставлять некоторые данные. Это может быть, например, наполнение справочников, новые системные настройки, демонстрационные записи раздела и т. д.

При создании раздела с помощью мастера к пакету автоматически привязываются данные, необходимые для регистрации и корректной работы раздела.



Привязать необходимые данные к пакету, содержащему разработанную функциональность, можно в разделе [*Конфигурация*] ([*Configuration*]).

Создать пользовательский пакет

Легкий

1. Создать пакет

1. Перейдите в дизайнер системы по кнопке .
2. В блоке [*Конфигурирование разработчиком*] ([*Admin area*]) перейдите по ссылке [*Управление конфигурацией*] ([*Advanced settings*]).

3. В области работы с пакетами нажмите кнопку .

2. Заполнить свойства пакета

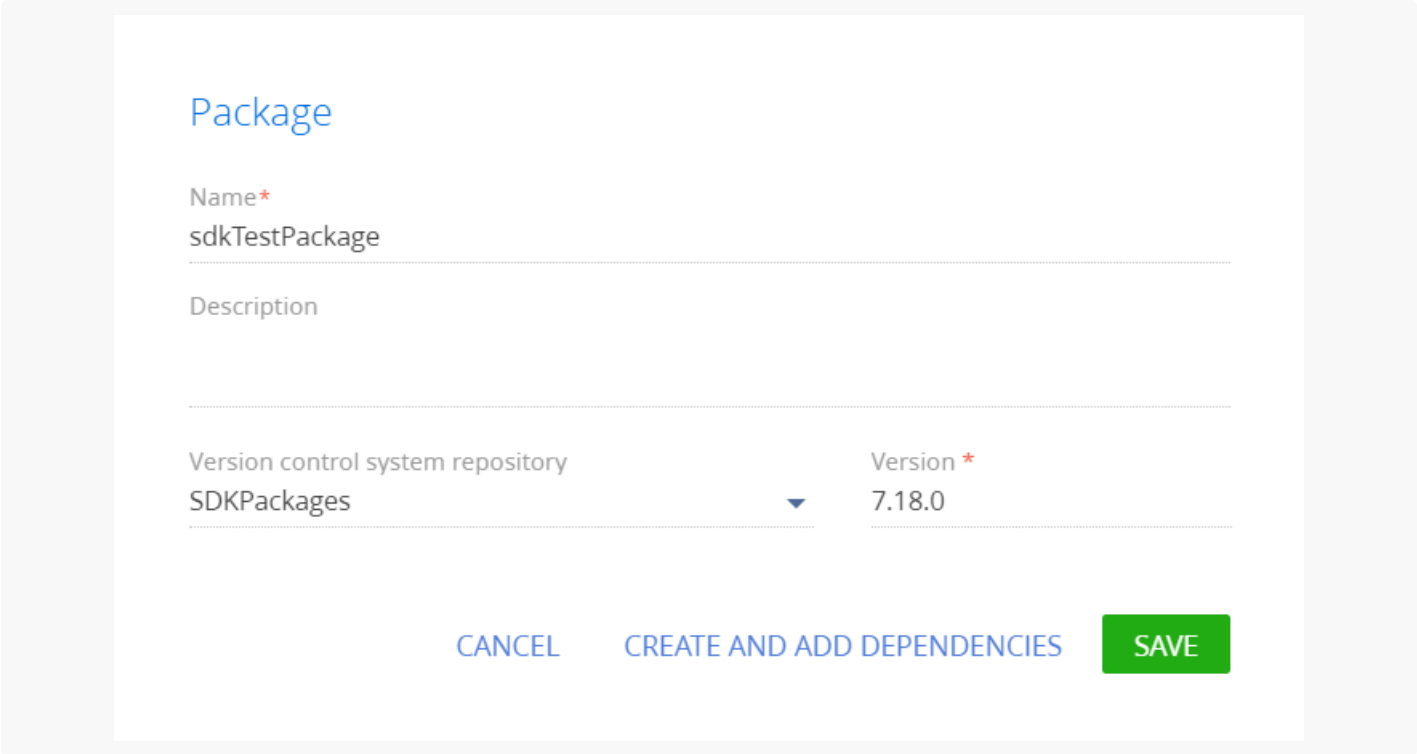
При нажатии на кнопку будет отображена карточка пакета, в которой необходимо заполнить свойства пакета.

Свойства пакета:

- [*Название*] ([*Name*]) — название пакета (обязательное свойство). Не может совпадать с названием существующих пакетов.
- [*Описание*] ([*Description*]) — описание пакета, например, расширенная информация о функциональности, которая будет реализована в пакете.
- [*Хранилище системы контроля версий*] ([*Version control system repository*]) — название хранилища системы контроля версий, в котором будут фиксироваться изменения пакета (обязательное свойство). Хранилища, которые находятся в перечне хранилищ конфигурации, но не помечены как активные, не попадут в выпадающий список доступных хранилищ.

  Важно. Поле [*Хранилище системы контроля версий*] ([*Version control system repository*]) заполняется при создании нового пакета и в дальнейшем недоступно для редактирования.

- [*Версия*] ([*Version*]) — версия пакета (обязательное свойство). Версия пакета может содержать цифры, символы латинского алфавита и знаки "." и "_". Добавляемое значение должно начинаться с цифры или буквы. Все элементы пакета имеют ту же версию, что и сам пакет. Указываемая версия пакета не обязательно должна совпадать с фактической версией приложения.



Содержимое свойств пакета будет сохранено в метаданных пакета.

Метаданные свойств пакета

```
{
    "Descriptor": {
        "UId": "1c1443d7-87df-4b48-bfb8-cc647755c4c1",
        "PackageVersion": "7.18.0",
        "Name": "NewPackage",
        "ModifiedOnUtc": "\/Date(1522657977000)\/",
        "Maintainer": "Customer",
        "DependsOn": []
    }
}
```

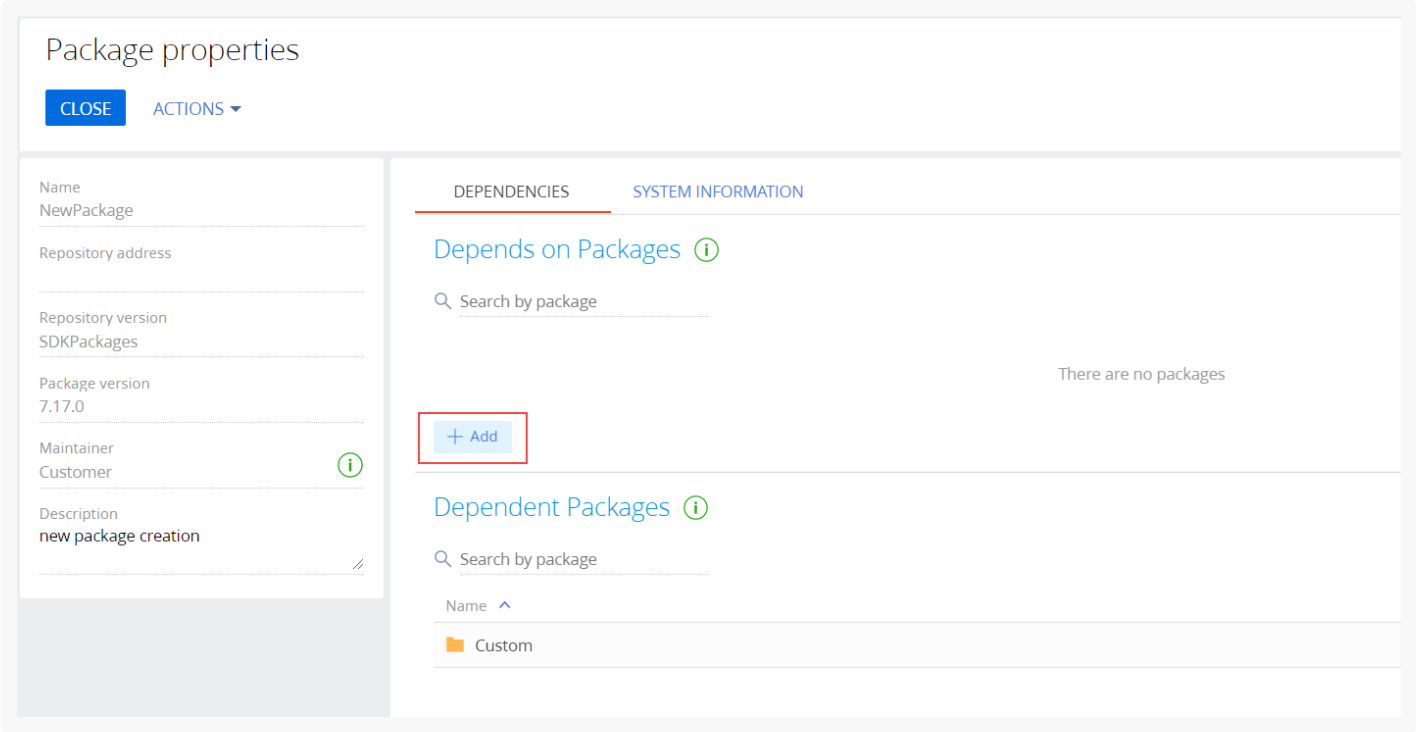
Кроме указанных выше свойств, метаданные пакета содержат информацию о зависимостях (свойство `DependsOn`) и информацию о разработчике (`Maintainer`). Значение свойства `Maintainer` устанавливается с помощью системной настройки [*Издатель*] (код `Maintainer`).

3. Определить зависимости пакета

Чтобы текущий пакет наследовал функциональность приложения, необходимо определить **зависимости пакета**.

Чтобы **добавить зависимости** пакета:

1. В карточке пакета нажмите кнопку [*Создать и добавить зависимости*] ([*Create and add dependencies*]).
2. На вкладке [*Зависимости*] ([*Dependencies*]) в детали [*Зависит от пакетов*] ([*Depends on packages*]) установите необходимые зависимости. Чтобы текущий пакет наследовал всю **функциональность приложения**, в качестве родительского пакета необходимо выбрать пакет, который в иерархии находится следующим после пакета [*Custom*].



4. Проверить зависимости пакета [*Custom*]

В пакете [*Custom*] должны быть установлены зависимости от всех пакетов приложения. Поэтому необходимо удостовериться в том, что в нем установлена зависимость от созданного пакета.

Привязать данные к пакету

Средний

Пример. Для пользовательского раздела [*Книги*] ([*Books*]) привязать демонстрационные записи и связанные с ними записи других разделов.

Демонстрационные записи:

- Книга David Flanagan "JavaScript: The Definitive Guide: Activate Your Web Pages", ISBN 978-0596805524, издательство "Apress", стоимость $33.89.
- Книга Andrew Troelsen "Pro C# 7: With .NET and .NET Core", ISBN 978-1484230176, издательство "Apress", стоимость $56.99.

1. Создать раздел

В нашем примере в мастере разделов предварительно был создан раздел [*Книги*] ([*Books*]). Поля раздела представлены в таблице.

Свойства колонок страницы записей раздела

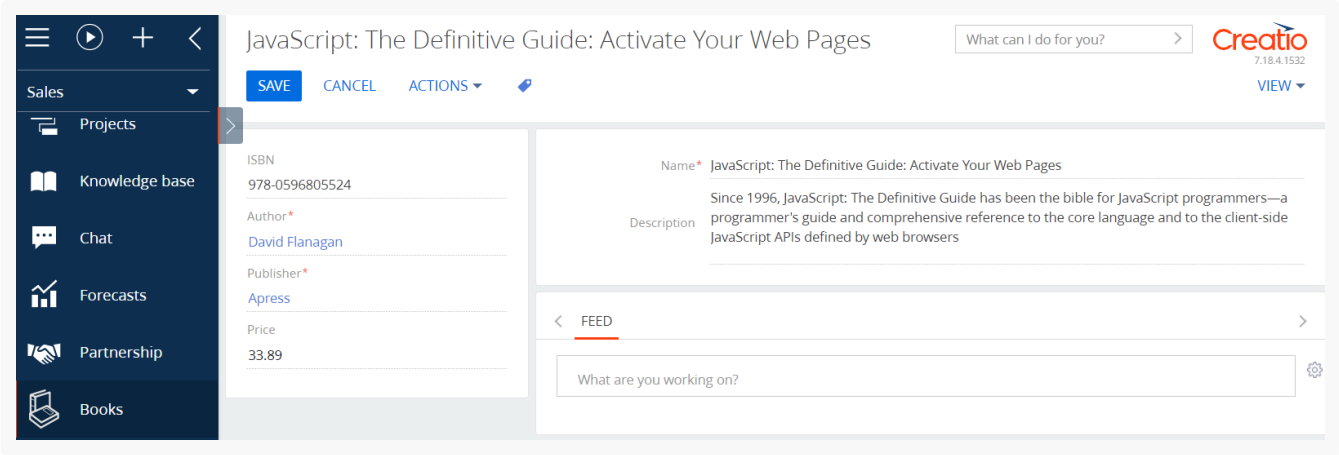
[*Заголовок*] ([*Title*])	**[*Код*] ([*Code*])**	**Тип данных**	**Обязательность поля**
[*Название*] ([*Name*])	`UsrName`	Строка (String)	Обязательное поле
[*ISBN*]	`UsrISBN`	Строка (String)	
[*Автор*] ([*Author*])	`UsrAuthor`	Справочник (Lookup) [*Контакт*] ([*Contact*])	Обязательное поле
[*Издатель*] ([*Publisher*])	`UsrPublisher`	Справочник (Lookup) [*Контрагент*] ([*Account*])	Обязательное поле
[*Стоимость*] ([*Price*])	`UsrPrice`	Дробное число (Decimal)	

Создание раздела подробно рассмотрено в статье Создать новый раздел.

2. Добавить в раздел демонстрационные записи

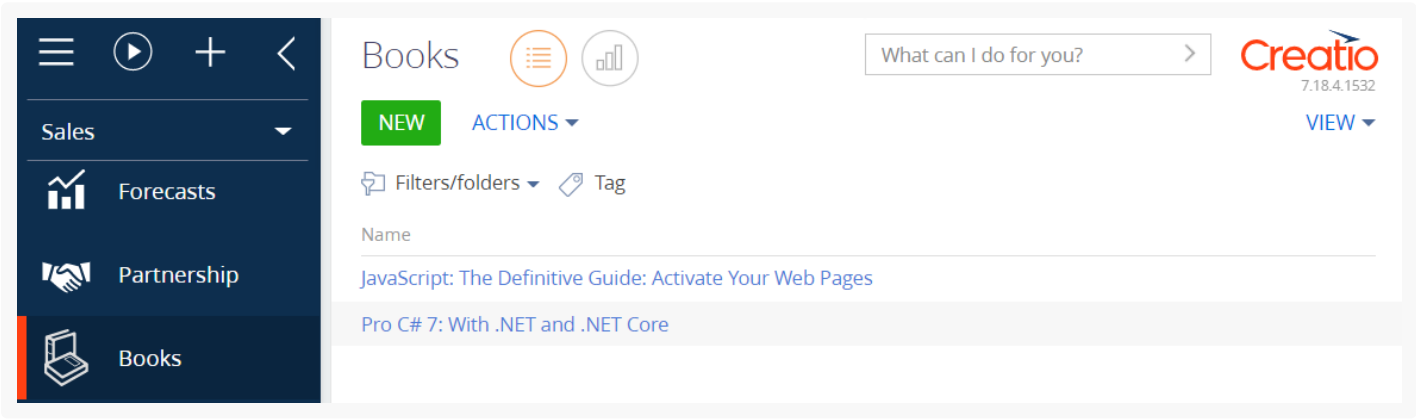
Чтобы **добавить записи** в реестр раздела [*Книги*] ([*Books*]):

1. В разделе [*Контакты*] ([*Contacts*]) добавьте запись и заполните поле [*ФИО*] ([*Full name*]) значением "`David Flanagan`".
2. В разделе [*Контакты*] ([*Contacts*]) добавьте запись и заполните поле [*ФИО*] ([*Full name*]) значением "`Andrew Troelsen`".
3. В разделе [*Контрагенты*] ([*Accounts*]) добавьте запись и заполните поле [*Название*] ([*Name*]) значением "`Apress`".
4. **Добавьте книгу** `JavaScript: The Definitive Guide: Activate Your Web Pages`:
   a. Перейдите в раздел [*Книги*] ([*Books*]).
   b. Нажмите [*Добавить*] ([*New*]).
   c. Заполните **поля** карточки книги:
      - [*Название*] ([*Name*]) — "`JavaScript: The Definitive Guide: Activate Your Web Pages`".
      - [*ISBN*] — "`978-0596805524`".
      - [*Автор*] ([*Author*]) — выберите "`David Flanagan`".
      - [*Издатель*] ([*Publisher*]) — выберите "`Apress`".
      - [*Стоимость*] ([*Price*]) — "`33.89`".



5. Аналогичным образом добавьте книгу `Pro C# 7: With .NET and .NET Core`.

Реестр раздела [*Книги*] ([*Books*]) представлен на рисунке ниже.



3. Привязать к пакету данные

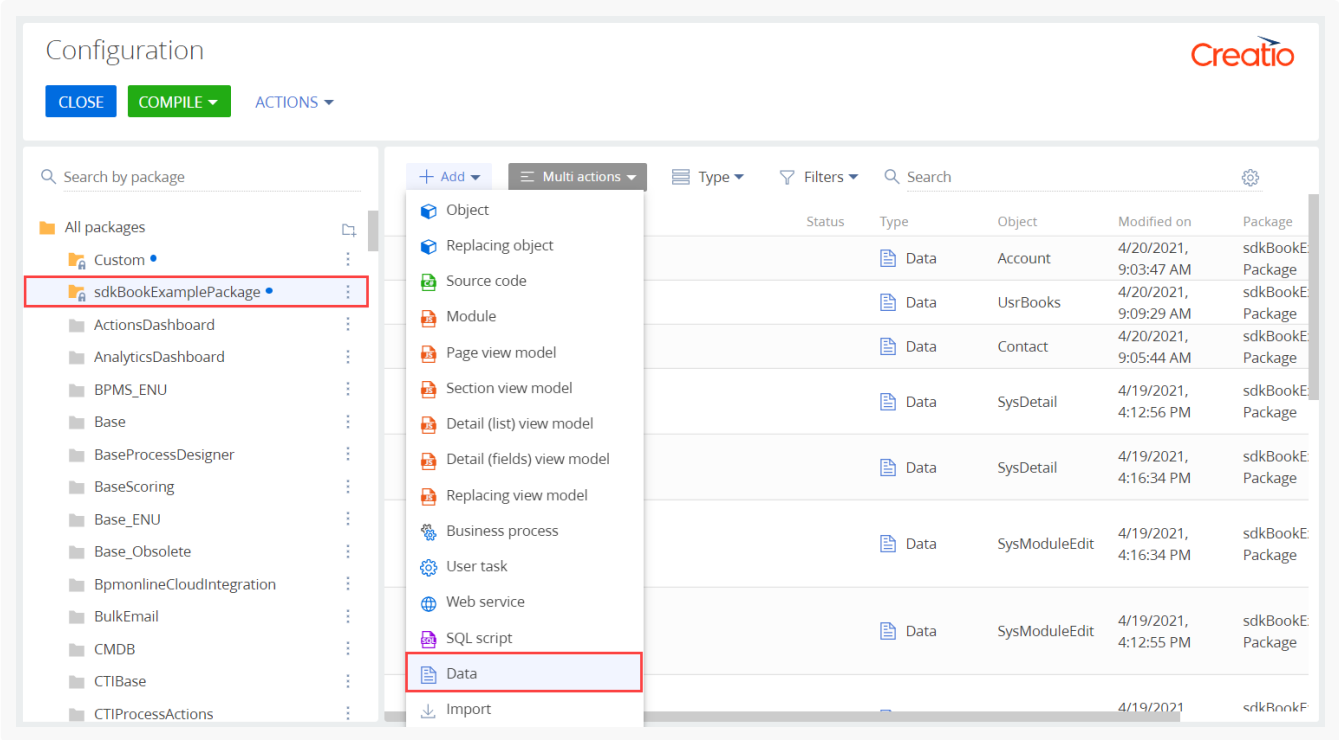
Поскольку записи раздела [*Книги*] ([*Books*]) связаны с записями раздела [*Контакты*] ([*Contacts*]) по колонке [*UsrAuthor*], то сначала необходимо привязать к пакету сведения об авторах.

Чтобы **выполнить привязку** данных к пакету:

1. Выполните **привязку контактов**:

   a. Перейдите в раздел [*Конфигурация*] ([*Configuration*]) и выберите пользовательский пакет.

   b. На панели инструментов рабочей области нажмите кнопку [*Добавить*] ([*Add*]) и выберите в списке вид конфигурационного элемента [*Данные*] ([*Data*]).



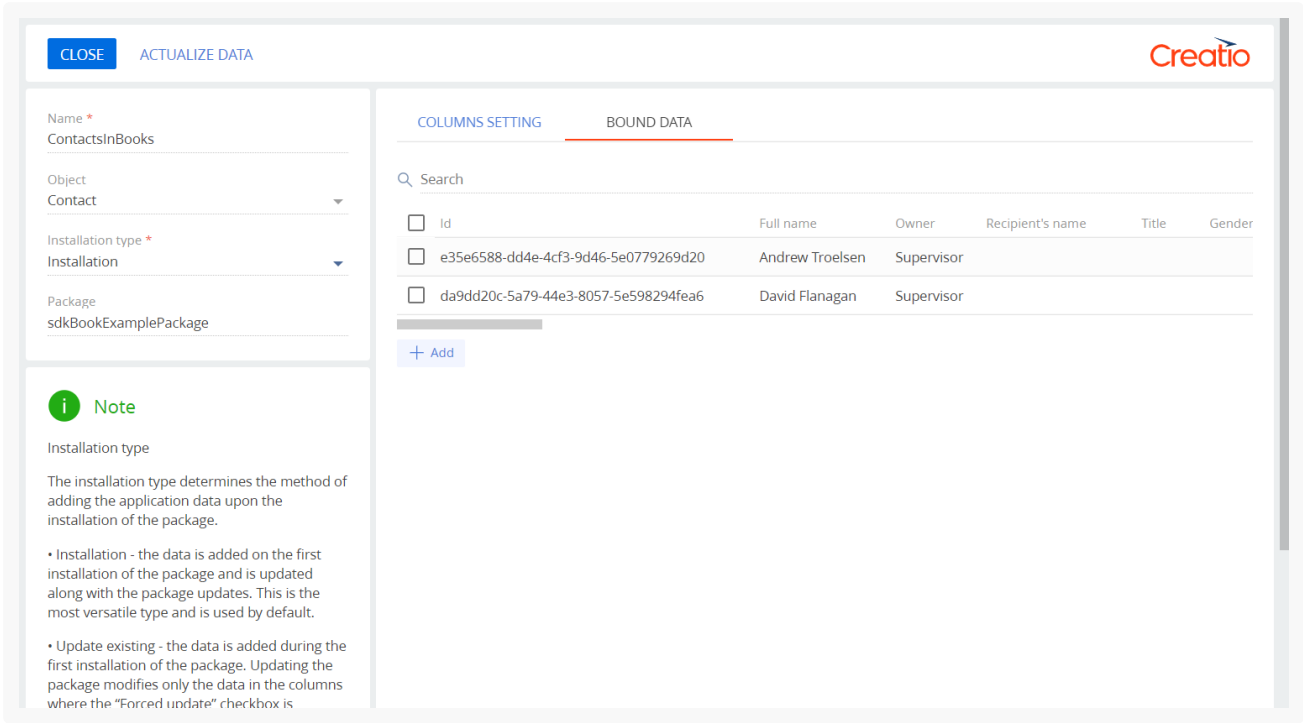
c. Заполните **свойства** страницы привязки данных:

   a. [*Название*] ([*Name*]) — "`ContactsInBooks`".

   b. [*Объект*] ([*Object*]) — "`Контакт`" ("`Contact`").

   c. [*Тип установки*] ([*Installation type*]) — "`Установка`" ("`Installation`").

   d. На вкладке [*Прикрепленные данные*] ([*Bound data*]) выберите записи, которые в колонке [*ФИО*] ([*Full name*]) содержат значения "`David Flanagan`" и "`Andrew Troelsen`".



e. Сохраните данные.

2. Выполните **привязку контрагента**:

   a. Перейдите в раздел [*Конфигурация*] ([*Configuration*]) и выберите пользовательский пакет.

   b. На панели инструментов рабочей области нажмите кнопку [*Добавить*] ([*Add*]) и выберите в списке вид конфигурационного элемента [*Данные*] ([*Data*]).

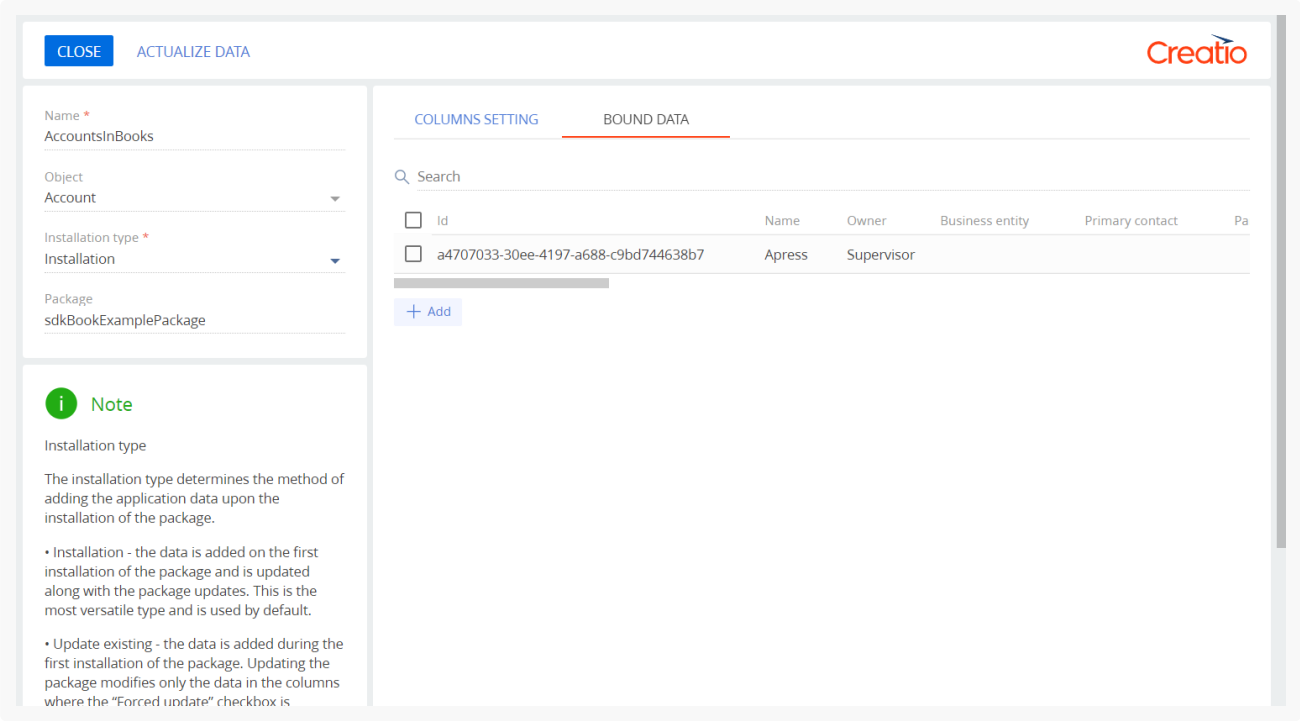
   c. Заполните **свойства** страницы привязки данных:

      a. [*Название*] ([*Name*]) — "`AccountsInBooks`".

      b. [*Объект*] ([*Object*]) — "`Контрагент`" ("`Account`").

      c. [*Тип установки*] ([*Installation type*]) — "`Установка`" ("`Installation`").

      d. На вкладке [*Прикрепленные данные*] ([*Bound data*]) выберите запись, которая в колонке [*Название*] ([*Name*]) содержит значение "`Apress`".



   e. Сохраните данные.

3. Выполните **привязку книг**:

   a. Перейдите в раздел [*Конфигурация*] ([*Configuration*]) и выберите пользовательский пакет.

   b. На панели инструментов рабочей области нажмите кнопку [*Добавить*] ([*Add*]) и выберите в списке вид конфигурационного элемента [*Данные*] ([*Data*]).

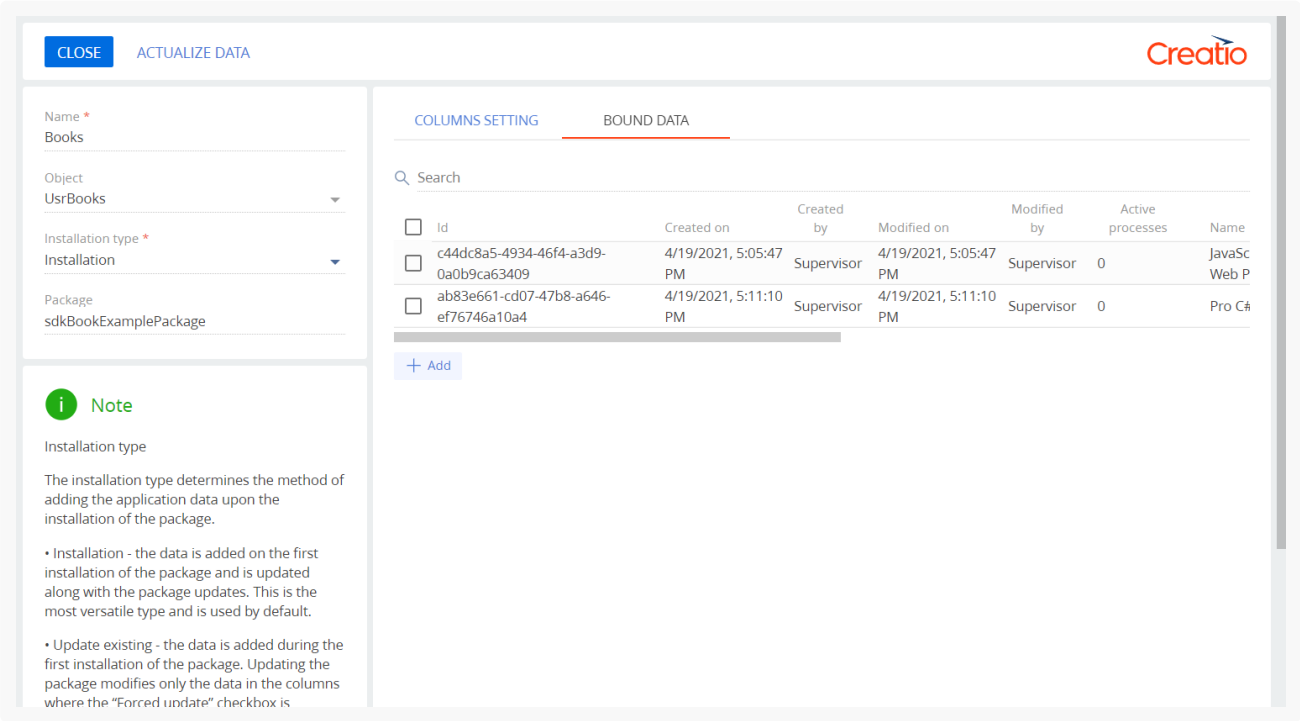
   c. Заполните **свойства** страницы привязки данных:

      a. [*Название*] ([*Name*]) — "`Books`".

      b. [*Объект*] ([*Object*]) — "`UsrBooks`".

      c. [*Тип установки*] ([*Installation type*]) — "`Установка`" ("`Installation`").

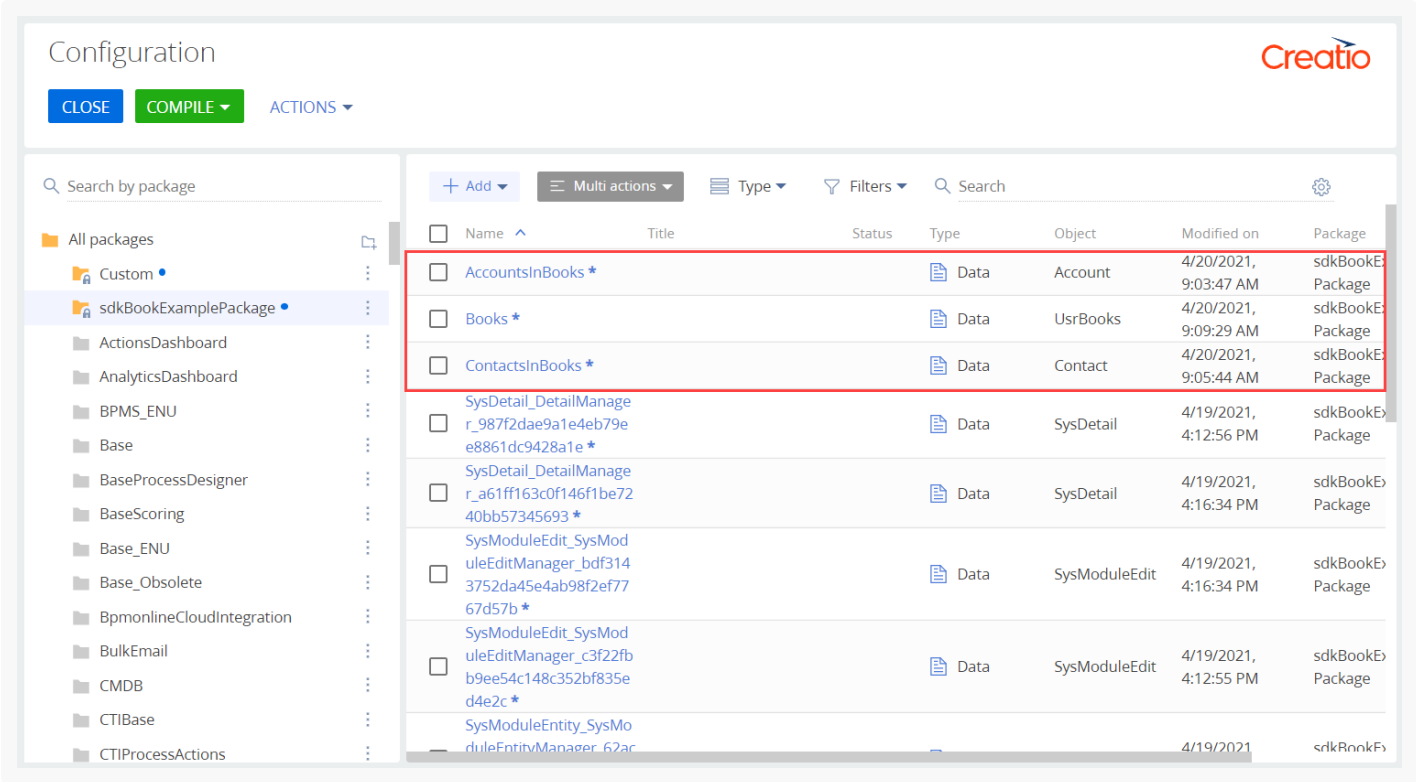
d. На вкладке [*Прикрепленные данные*] ([*Bound data*]) выберите записи, которые в колонке [*Название*] ([*Name*]) содержат значения "`JavaScript: The Definitive Guide: Activate Your Web Pages`" и "`Pro C# 7: With .NET and .NET Core`".



e. Сохраните данные.

4. Проверить привязки данных

В результате выполнения примера к пользовательскому пакету будут привязаны данные разделов "[*Книги*]" ("[*Books*]"), "[*Контакты*]" ("[*Contacts*]"), "[*Контрагенты*]" ("[*Accounts*]").



Теперь пакет полностью готов для переноса между рабочими средами с помощью механизма экспорта и импорта пакетов Creatio IDE. После установки пакета в другую рабочую среду все привязанные записи отобразятся в соответствующих разделах.