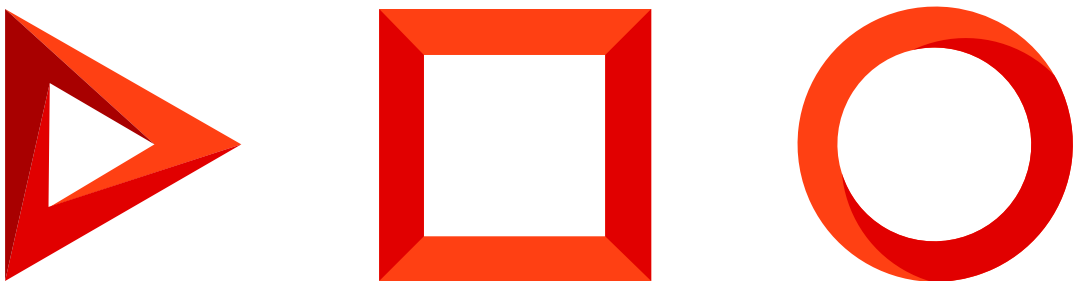
Creatio Academy

# Операции с данными (back-end)

## Копирование иерархических данных

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

# Копирование иерархических данных

Сложный

Для копирования записей таблицы базы данных и записей связанных таблиц используется **копирование иерархических данных**. В Creatio иерархическое копирование данных доступно к использованию для копирования таблиц `[ProductHierarchyDataStructureObtainer]` и `[ProductConditionHierarchyDataStructureObtainer]`. Чтобы скопировать данные с других таблиц, необходимо выполнить кастомизацию иерархического копирования данных.

Иерархическое копирование можно использовать в пользовательском веб-сервисе, например, при копировании данных из внешнего сервиса в приложение Creatio или при подключении к внешней базе данных.

Например, есть раздел [ *Продукты* ] ([ *Products* ]), который сформирован на основе таблицы `[Product]` базы данных. Страница продукта содержит пользовательские детали. При иерархическом копировании записи раздела будут скопированы и данные записи, и данные связанных деталей.

При копировании учитываются права доступа к таблицам. Например, пользователь не имеет прав на чтение и добавление записей в таблицу `[Contact]`. При вызове копирования иерархических данных запись не будет скопирована и приложение вернет сообщение о невозможности выполнения данной операции.

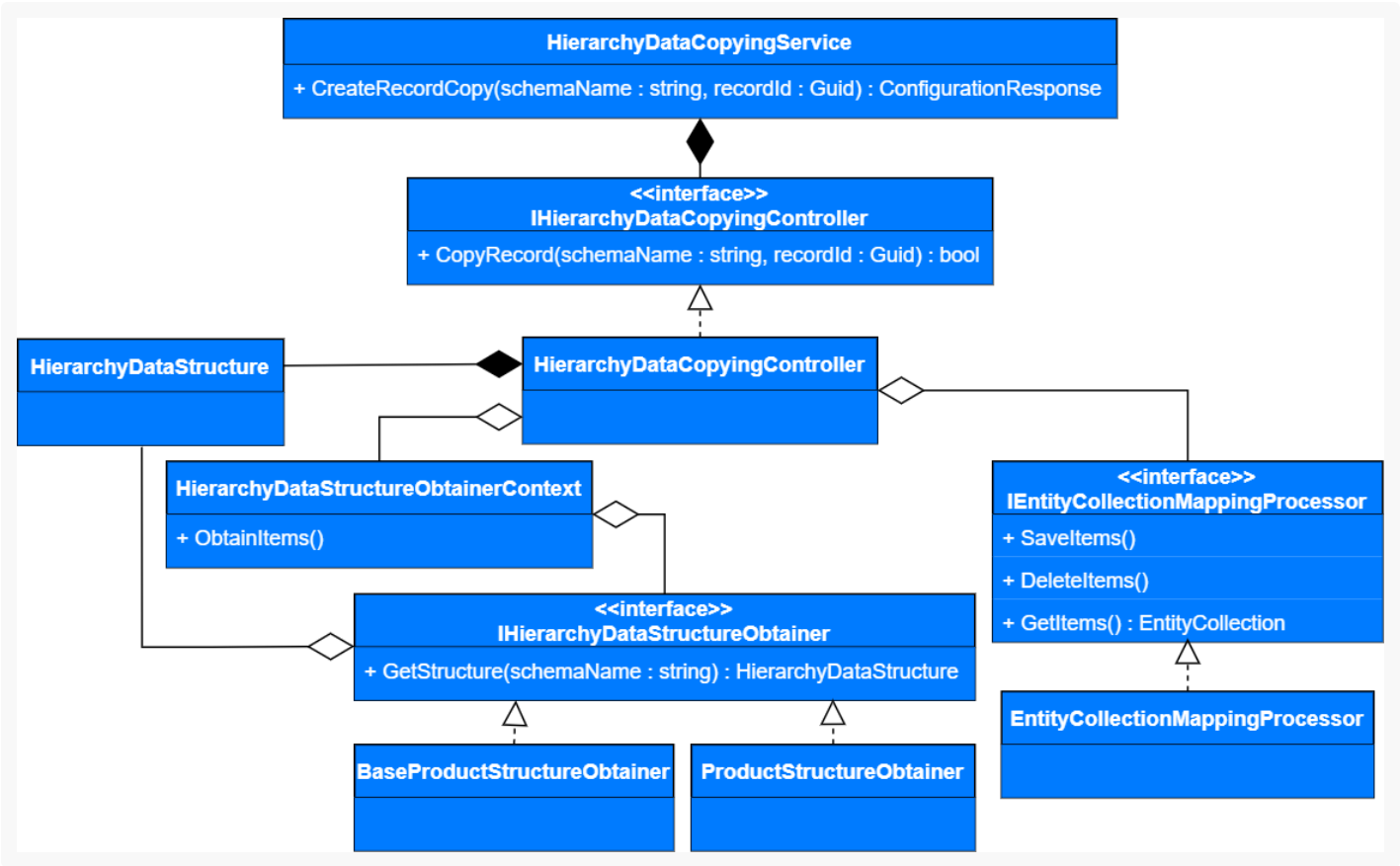
## Структура и алгоритм работы копирования иерархических данных

**Составляющие** копирования иерархических данных представлены в таблице.

Составляющие копирования иерархических данных

| Название | Описание | Классы и интерфейсы |
|---|---|---|
| **Контроллер** | Контролирует процесс копирования | `IHierarchyDataCopyingController`<br>`HierarchyDataCopyingController` — контроль создания копии записи таблицы и связанных данных из базы данных. |
| **Получатель** | Получает из базы данных структуру текущей таблицы и связанных таблиц | `HierarchyDataStructureObtainerContext` — выбор алгоритма для получения иерархической структуры таблицы и связанных таблиц.<br>`IHierarchyDataStructureObtainer`<br>`HierarchyDataStructureObtainer` — получение иерархической структуры таблицы и связанных таблиц.<br>`BaseHierarchyDataStructureObtainer`<br>`ProductHierarchyDataStructureObtainer` — получение иерархической структуры таблицы `[Product]` и связанных таблиц. |
| **Контейнер** | Сохраняет структуру текущей таблицы и связанных таблиц | `HierarchyDataStructure` — контейнер для сохранения информации о структуре иерархических данных. |
| **Мапер** | Работает со структурой | `IEntityCollectionMappingProcessor`<br>`EntityCollectionMappingProcessor` — получение структуры из базы данных, копирование. |

**Диаграмма классов** копирования иерархических данных представлено на рисунке ниже.




**Алгоритм работы** копирования иерархических данных:

1. Класс сервиса вызывает контроллер процесса копирования и передает в него название таблицы и идентификатор записи, данные которой будут копироваться.
2. Контроллер начинает поэтапное создание копии:
   a. Получает структуру таблицы и связанных таблиц в унифицированной форме.
   b. Сохраняет полученную структуру таблицы в унифицированной форме.
3. Контроллер копирует записи в соответствии со структурой, полученной на этапе сохранения.

**Унифицированная форма** — это сохранение полученной структуры таблицы в объект с типом `HierarchyDataStructure`. Если таблица имеет связанные таблицы (колонка по внешнему ключу ссылается на запись другой таблицы), то они помещаются в созданный объект (в коллекцию из объектов аналогичного типа). При необходимости расширения или обновления механизма получения структуры использование унифицированной формы позволяет обработать новую структуру без дополнительных изменений кода в контроллере.

Шаблон унифицированной формы, которая используется при сохранении полученной структуры таблицы, приведен ниже.

**Шаблон унифицированной формы структуры таблицы**

```
/* Class holds structure of hierarchical data. */
public class HierarchyDataStructure
```

```
{
    public string SchemaName;
    public List<string> Columns;

    /* If current structure object does not have a parent foreign table name than here need to b
    public string ParentColumnName;

    /* List of child structures. */
    public List<HierarchyDataStructure> Structures;

    /* List filters. */
    public HierarchyDataStructureFilterGroup Filters;
}
```

# Кастомизировать копирование иерархических данных

Кастомизация копирования иерархических данных позволяет:

- Добавить пользовательскую реализацию получателя данных (класс `HierarchyDataStructureObtainer`).
- Изменить реализацию получателя данных (класс `HierarchyDataStructureObtainer`).
- Изменить реализацию контроллера (класс `HierarchyDataCopyingController`).
- Добавить пользовательскую реализацию копирования иерархических данных.

## Добавить пользовательскую реализацию получателя данных

**Способы** добавления пользовательской реализации получателя данных (класс `HierarchyDataStructureObtainer`):

- Через базовый интерфейс.
- Через наследование базового класса.

### Добавить пользовательскую реализацию получателя данных через базовый интерфейс

1. Создайте класс, который реализует интерфейс `IHierarchyDataStructureObtainer`. Шаблон названия класса: `[НазваниеОбъектаКопирования]HierarchyDataStructureObtainer`.
2. Добавьте пользовательскую реализацию метода интерфейса `ObtainStructure()`. Обязательно укажите модификатор `virtual`.

Пример реализации получателя данных через базовый интерфейс содержится в пакете [ *ProductBankCustomerJourney* ] —> классы `ProductHierarchyDataStructureObtainer` и `ProductConditionHierarchyDataStructureObtainer`.

### Добавить пользовательскую реализацию получателя данных через наследование

### базового класса

Под базовой реализацией необходимо понимать стандартное копирование записи без связанных записей. Получатель реализован в классе `BaseHierarchyDataStructureObtainer` базового пакета [ *NUI* ].

1. Создайте класс, который реализует интерфейс `BaseHierarchyDataStructureObtainer` (пакет [ *NUI* ] —> класс `BaseHierarchyDataStructureObtainer`). Шаблон названия класса: `[НазваниеОбъектаКопирования]HierarchyDataStructureObtainer`.
2. Расширьте базовую реализацию получателя.

Пример реализации получателя данных через наследование базового класса содержится в пакете [ *ProductBankCustomerJourney* ] —> класс `ProductHierarchyDataStructureObtainer`).

## Изменить реализацию получателя данных

1. Создайте класс, который замещает один из классов `BaseHierarchyDataStructureObtainer` (пакет [ *NUI* ]), `ProductConditionHierarchyDataStructureObtainer` (пакет [ *ProductBankCustomerJourney* ]), `ProductHierarchyDataStructureObtainer` (пакет [ *ProductBankCustomerJourney* ]).
2. В замещающий класс добавьте пользовательскую реализацию замещающего метода `ObtainStructure()` базового класса.

## Изменить реализацию контроллера

1. Создайте класс, который реализует интерфейс `IHierarchyDataCopyingController`. Шаблон названия класса: `[НазваниеОбъекта]HierarchyDataController`.
2. В метод интерфейса `CopyRecord` добавьте пользовательский алгоритм копирования.

   Один шаг алгоритма должен содержать вызов одного метода другого класса. Шаг алгоритма также должен включать в себя создание объекта класса, который необходимо вызвать, или минимальную подготовку данных, которые будут передаваться в метод.

## Добавить пользовательскую реализацию копирования иерархических данных

1. Добавьте реализацию контроллера процесса копирования (класс `HierarchyDataCopyingController`). Контроллер должен поэтапно вызывать получателя структуры (класс `HierarchyDataStructureObtainer`), обработчика структуры (класс `EntityCollectionMappingProcessor`), контейнера структуры (класс `HierarchyDataStructure`).
2. Добавьте реализацию получателя иерархической структуры данных (класс `HierarchyDataStructureObtainer`).
3. Создайте класс, который реализует интерфейс. Шаблон названия класса: `[НазваниеОбъекта]HierarchyDataProcessor`.

   Рекомендуется добавить интерфейс для класса обработчика. Это позволяет добавить другую реализацию и заменить существующую, а также используется для унификации всех обработчиков.
4. Создайте класс, реализующий интерфейс `IEntityCollectionMappingHandler`.

5. В метод контроллера `CopyRecord` добавьте вызовы методов получателя структуры (класс `HierarchyDataStructureObtainer`), обработчика структуры (класс `EntityCollectionMappingProcessor`), контейнера структуры (класс `HierarchyDataStructure`).
6. Создайте в пользовательском классе объект класса `HierarchyDataCopyingController`.

**Пример создания объекта контроллера**

```
var copyController = ClassFactory.Get<HierarchyDataCopyingController>(new ConstructorArgument
```

7. Вызовите метод копирования `copyController`.

**Пример вызова метода копирования**

```
copyController.CopyRecord(schemaName, recordId);
```

# Вызвать копирование иерархических данных

Копирование иерархических данных можно вызвать из front-end и из back-end части.

## Вызвать иерархическое копирование из front-end части

Чтобы **вызвать копирование иерархических данных из front-end части**, используйте метод `callService()`.

Пример вызова содержится в пакете [ *ProductBankCustomerJourney* ] —> схема `ProductConditionDetailV2` —> метод `callCopyRecordService()`.

**Пример вызова сервиса копирования из front-end части**

```
/**
* Call service that creates records copy.
* @protected
*/
callCopyRecordService: function() {
    this.showBodyMask();
    var config = this.getCopyRecordConfig();
    this.callService(config, this.copyRecordServiceCallback, this);
}
```

## Вызвать иерархическое копирование из back-end части

Чтобы **вызвать копирование иерархических данных из back-end части**, в пользовательском классе создайте объект класса `HierarchyDataCopyingController`.

**Пример вызова сервиса копирования из back-end части**

```
var copyController = ClassFactory.Get<HierarchyDataCopyingController>(new ConstructorArgument("U
```

Чтобы **работать с данными таблиц по маппингу колонок**:

1. В пользовательском классе создайте объект класса маппера, который реализует интерфейс `IEntityCollectionMappingHandler`.

   **Пример создания объекта класса маппера**

   ```
   var entityCollectionMappingHandler = ClassFactory.Get<IEntityCollectionMappingHandler>(new Co
   ```

2. Вызовите методы маппера через объект.

   **Пример вызова метода копирования**

   ```
   entityCollectionMappingHandler.CopyItems(data.SchemaName, columns, filterGroup, relatedColumn
   ```

   **На заметку.** Создание объекта по названию интерфейса позволяет разработчику заменить существующую реализацию маппера на пользовательскую. При изменении реализации существующего маппера будет перекомпилирован только класс маппера. Все классы, которые используют данную реализацию, не требуют перекомпиляции. Подробнее о создании объектов с использованием механизма внедрения зависимостей описано в статье Замещение конфигурационных элементов.

Чтобы **получить структуру определенной таблицы** в виде объекта с типом `HierarchyDataStructure`:

1. В пользовательском классе создайте объект класса `HierarchyDataStructureObtainerContext`.

   **Пример создания получателя структуры таблицы**

   ```
   var _hierarchyDataStructureObtainer = ClassFactory.Get<HierarchyDataStructureObtainerContext>
   ```

2. Получите структуру определенной таблицы.

   **Способы** получения структуры:

   - Вызовите метод `ObtainStructureByObtainerStrategy` и передайте ему параметр `schemaName` — название таблицы, запись которой необходимо скопировать.
   - Вызовите реализацию существующего получателя структуры — `ProductHierarchyDataStructureObtainer` или `ProductConditionHierarchyDataStructureObtainer`.