EXT: Внешний импорт

Ключ расширения: external_import

Язык: ru

Ключевые слова: forAdmins, forIntermediates

Содержание

Введение

This extension is designed to fetch data from external sources and store them into tables of the TYPO3 database. The mapping between this external data and the TYPO3 tables is done by extending the syntax of the TCA. A backend module provides a way to synchronise any table manually or to defined a scheduling for all synchronisations. Automatic scheduling relies on Gabriel (TYPO3 4.2 or less) or the Scheduler (TYPO3 4.3+).

The main idea of getting external data into the TYPO3 database is to be able to use TYPO3 standard functions on that data (such as enable fields, for example, if available).

Connection to external applications is handled by a class of services called "connectors", the base of which is available as a separate extension (svconnector).

Data from several external sources can be stored into the same table allowing data aggregation.

The extension also provides an API for sending it data from some other source. This data is stored into the TYPO3 database using the same mapping process as when data is fetched directly by the extension.

This extension contains a number of hooks as well as the possibility to call user-defined functions during the import process, which makes it a quite flexible tool. However it was not designed for extensive data manipulation. It is assumed that the data received from the external source is in "palatable" format. If the external data requires a lot of processing, it is probably better to put it through an ETL or ESB tool first, and then import it into TYPO3.

Please also check extension "external_import_tut" which provides a tutorial to this extension.

Вопросы и поддержка

О своей проблеме можно написать в английском списке рассылки TYPO3 (typo3.english), чтобы и другие могли извлечь пользу из ответов. Проблемы и предложения направляйте в виде записей в разделе отслеживания ошибок расширения в Forge (http://forge.typo3.org/projects/extension-external_import/issues).

Если разработчики счастливы...

Если расширение понравилось Вам, не забудьте оценить его. Перейдите в репозиторий расширений, найдите это расширение, щелкните по его названию и перейдите к детальному представлению, где щелкните по вкладке "Ratings" и дайте оценку (необходима авторизация). Каждый голос засчитывается в пользу разработчику. Не забывайте об этом!

Кроме того, можно вернуться назад и подумать о пользе обмена. Подумайте о том, сколько пользы Вы принесли сообществу и сколько оно дало Вам.

Соучастие

Этот инструмент может использоваться в самых разных ситуациях, которые не могут быть охвачены текущей версией. Вероятно мне не хватит времени на реализацию не нужных лично мне вариантов. Но Вы можете влиться в команду разработчиков для привнесения новых возможностей. Если интересно, перейдите на forge.typo3.org и отправьте запрос на вступление в команду разработчиков данного расширения. Мы свяжемся с Вами.

Установка

Сама по себе установка этого расширения ничего не делает. Кроме того понадобится дополнить определение TCA некоторых таблиц с соответствующим синтаксисом и создать специфические соединения для определенных, нужных вам приложений.

Установка расширения Gabriel не обязательна, но требуется для определения расписания автоматической синхронизации. Обратите внимание, что необходима последняя версия Gabriel, а не та, что доступна в TER. Последнюю версию Gabriel можно получить на forge: http://forge.typo3.org/projects/show/extension-gabriel.

Если используется версия TYPO3 4.3 или выше, вместо Gabriel можно использовать системное расширение Scheduler.

Замечания по совместимости

Обновление до TYPO3 4.3 и Планировщик

Если уже имеется полностью настроенное Gabriel на TYPO3 4.2 или менее, процесс обновления не пройдет абсолютно гладко. Несмотря на то, что TYPO3 4.3 имеет интегрированный в ядро Gabriel под названием "Scheduler". Это системное расширение, представляющее серьезно исправленный Gabriel.

Поэтому, если происходит обновление до TYPO3 4.3 и выше, необходимо отказаться от Gabriel и вместо него использовать Scheduler. Сложность в том, что настроенные задачи будут потеряны, так как их невозможно преобразовать между Gabriel и Scheduler (слишком сильные отличия в двух инструментах). Это не должно остановить Вас, так как Scheduler дает больший контроль и отчетность над запланированными задачами (а поддержка Gabriel может отказаться от внешнего импорта в будущем).

Обновление до 0.8.0

Начиная с версии 0.8.0 стало возможным определить несколько внешних источников для заданной таблицы. Это достигается через изменившийся расширенный синтаксис TCA. При обновлении на версию 0.8.0 необходимо кроме того изменить все свои "external" свойства TCA. Все эти свойства стали индексными массивами. Поэтому, если было следующее:

```
$TCA['tx_myext_mytable'] = array (
      'ctrl' => array (
            'title'     => ...,
            ...
            'external' => array(
                  'connector' => ...,
                  'parameters' => array(
                        ...
                  ),
                  'data' => 'xml',
                  'nodetype' => 'record',
                  'reference_uid' => ...,
                  'priority' => 10,
                  'deleteNonSynchedRecords' => 1
            )
      ),
);
```

То необходимо поменять на это:

```
$TCA['tx_myext_mytable'] = array (
      'ctrl' => array (
            'title'     => ...,
            ...
            'external' => array(
                  0 => array (
                        'connector' => ...,
                        'parameters' => array(
                          ...
                        ),
                        'data' => 'xml',
                        'nodetype' => 'record',
                        'reference_uid' => ...,
                        'priority' => 10,
                        'deleteNonSynchedRecords' => 1
                  )
            )
      ),
);
```

То же самое и с определением столбцов, которое изменилось с:

```
'field_name' => array (
      'exclude' => 0,
```

```
      'label' => '...',
      'config' => array (
            ...
      ),
      'external' => array (
            'field' => '...',
      )
),
```

на:

```
'field_name' => array (
      'exclude' => 0,
      'label' => '...',
      'config' => array (
            ...
      ),
      'external' => array (
            0 => array (
                  'field' => '...',
            )
      )
),
```

Кроме того, синтаксис разметки MM был упрощен. И следующая настройка:

```
'external' => array(
      0 => array(
            'MM' => array(
                  'mappings' => array(
                        'uid_foreign' => array(
                          'table' => name of foreign table,
                          'reference_field' => foreign MM key,
                          'value_field' => 'uid'
                        )
                  ),
                  'additional_fields' => array(
                        TYPO3 field name => external data field name
                  ),
                  'sorting' => 'field',
            )
      )
)
```

может быть перезаписана так:

```
'external' => array(
      0 => array(
            'MM' => array(
                  'mapping' => array(
                        'table' => name of foreign table,
                        'reference_field' => foreign MM key,
                        'value_field' => 'uid'
                  ),
                  'additional_fields' => array(
                        TYPO3 field name => external data field name
                  ),
                  'sorting' => 'field',
            )
      )
)
```

Хотя старый синтаксис также работает.

Кроме того, обратите внимание, что свойство "deleteNonSynchedRecords" устарело ввиду появления более гибкого свойства "disabledOperations" (смотрите Настройки ниже). Хотя оно все еще поддерживается.

Ожидается, что это были последние важные изменения в синтаксисе, поэтому статус расширения был повышен до бетта.

Обновление с 0.5.0

Если Вы использовали версию 0.5.0, то можете удивиться некоторым сюрпризам, так как расширенный синтаксис TCA для MM связей был изменен:

- В разметке MM, разметку "uid_local" более определять не требуется. Действительно, локальные uid всегда будут "uid", так как весь смысл этого расширения — сохранять данные в таблицах базы данных, соответствующим стандартам TYPO3.
- "reference_field" для разметки "uid_foreign" теперь использует названия полей в таблице локальной базы данных. Оно соответствует названию поля внешних данных, читаемого из соответствующего поля.

- Свойство “update” было удалено, хотя TCEmain в любом случае удаляет существующие связи MM.
- Поле “sorting_data” было удалено. Свойство “sorting” теперь хранить то, что было в “sorting_data”, а других параметров для сортировки не существует.

Другие требования

Как уже говорилось во введении, это расширение интенсивно использует расширенный синтаксис TCA. Если Вы не знакомы с TCA, то настойчиво рекомендуем сначала прочитать документацию **Core API**.

Настройки

Расширение имеет следующие параметры настройки:

- **Storage PID**: определение страницы, хранящей все импортированные записи. Для каждой таблицы может быть переназначено (смотрите "Управление" ниже).
- **Force PHP time limit**: установка максимального времени выполнения, отличающегося от установки по умолчанию. Установка на -1 сохраняет ограничение времени по умолчанию. Установка количества секунд изменяет максимальное время выполнения. Полезно при импорте данных большого размера.
- **Email for reporting**: если ввести здесь адрес email, то после каждой автоматической синхронизации на него будет высылаться детальный отчет. После ручной синхронизации из внутреннего модуля письма не отсылаются.
- **Subject of email report**: приставка к теме письма-отчета. Полезно, например, использовать название сервера, если один и тот же импорт выполняется на нескольких серверах.
- **Preview/Debug limit**: максимальное количество строк, сбрасываемых в devlog при отключении отладки. Также используется в качестве количества строк, отображаемых при предварительном просмотре, если это применимо.
- **Debug**: включите для вывода некой отладочной информации (требуется расширение, подобное devlog).
- **Disable logging**: включите для отключения журналирования в TCEmain. По умолчанию ведется запись в журнал TYPO3 (который просматривается через модуль Сервис управления > Журнал) для каждой записи, участвующей в процессе импорта. Количество записей при этом может быть огромным, поэтому иногда полезно отключать журналирования процесса импорта, если он потом никак не отслеживается. Включение этого флажка отключает журналирование для **всех** таблиц. Может быть переназначено на уровне таблиц через флаг "disableLog" (смотрите "Общая настройка TCA").
- **Clean AJAX output**: этот параметр **включен по умолчанию**. При загрузке синхронизации через внутренний модуль, сценарий синхронизации вызывается через AJAX. Этот сценарий может вызвать ошибки PHP или любого рода отладочную информацию (например, ошибки запросов SQL при включении `SQLDebug`). Это может прервать ответ AJAX и отображение модуля внутреннего интерфейса. При включении этого параметра, все выводимое внешним импортом будет вычищаться, делая возможным получение чистого ответа AJAX, расплачиваться придется потерей сообщений о возможных ошибках. За деталями смотрите "Отладка".
  Замечание: параметр игнорируется начиная с TYPO3 4.3, так как выводимое там в любом случае очищается. Недостаток в том, что в TYPO3 4.3 и выше никогда нельзя увидеть ошибки.

Руководство пользователя

Общие положения

Цель этого расширения — захват данных извне (назовем это внешним источником) локальной базы данных TYPO3, и сохранение их в локальной базе данных. Данные из внешних источников соотносятся с локальными таблицами и полями, в соответствие с информацией, хранящейся в TCA, использующей расширенный синтаксис, обеспеченный этим расширением.

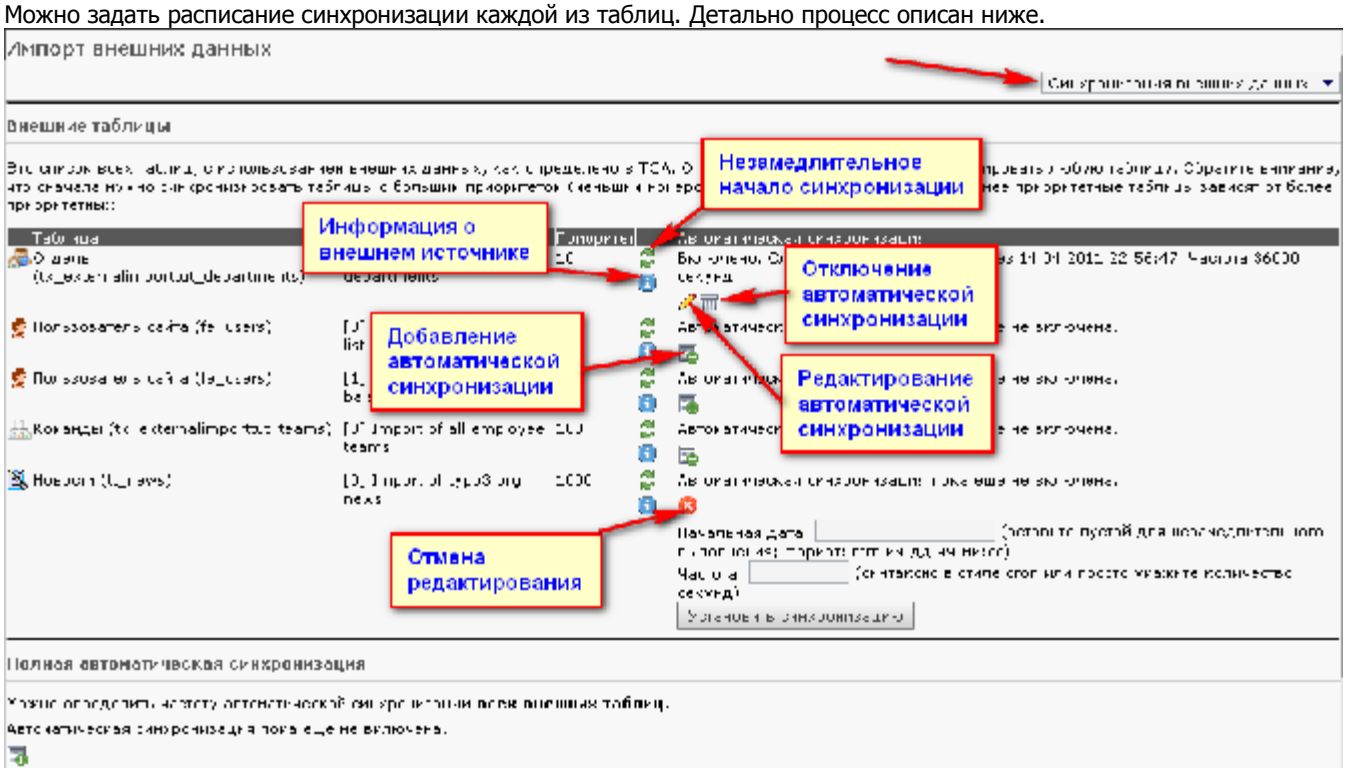
Расширение может либо получать данные из некого внешнего источника, либо получать данные из любого возможного сценария. Получение данных из внешнего источника происходит при помощи стандартного процесса. Соединение со внешними источниками происходит через службы соединений (смотрите расширение "svconnector"), которые возвращают полученные данные во внешний импорт. Если такое соединение существует, оно может быть связано с одной или несколькими таблицами TYPO3 (при необходимости, с дополнительными параметрами) через расширенный синтаксис TCA. После чего таблица может быть синхронизирована с внешним источником. При начале каждой синхронизации (в ручную, либо по расписанию), для получения данных вызывается служба соединения. Такие таблицы называются здесь "синхронизируемые". Такой тип действий называется "извлечение данных".

С другой стороны, расширение предоставляет интерфейс (API), который может быть вызван для передачи данных непосредственно во внешний импорт. Службы соединений при этом не используются. Расширение вызывается по необходимости любым использующим его сценарием. При это невозможно ни синхронизировать таблицы из внутреннего модуля, ни установить расписание синхронизации. Такие таблицы называются "не синхронизируемые". Такой тип действий называется "получение данных".

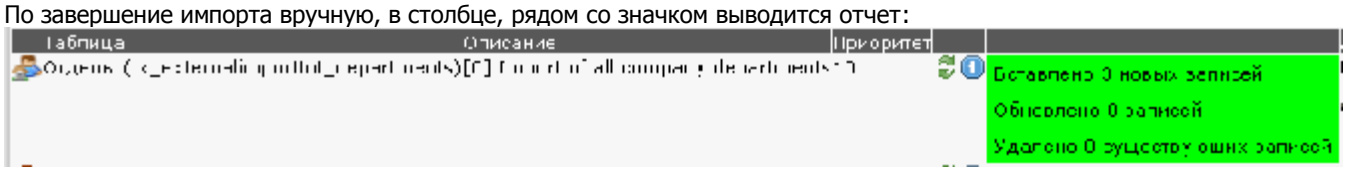
Обратите внимание, что вполне возможно получать данные и для синхронизируемых таблиц. Обратное неверно (не синхронизируемые таблицы не могут извлекать данные).

Синхронизируемые таблицы

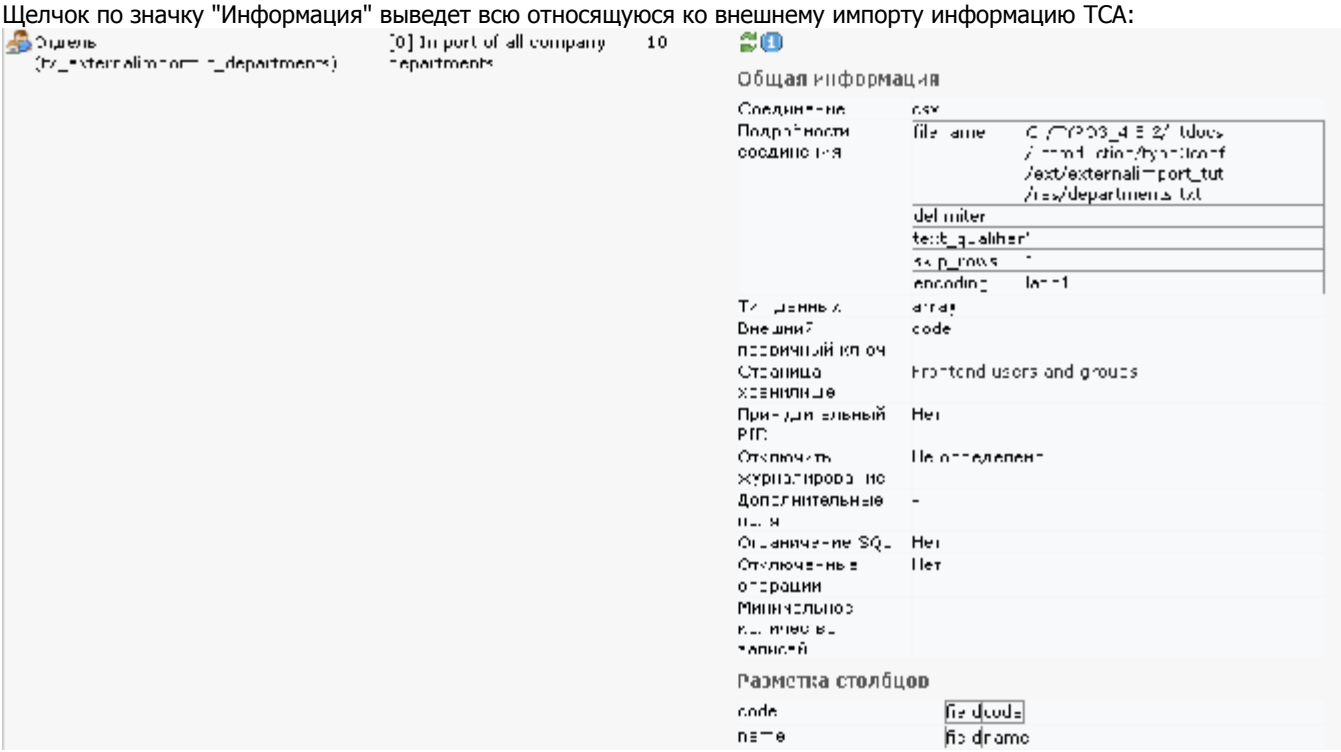
Первая функция внутреннего модуля под названием "Синхронизация внешних данных" отображает список синхронизируемых таблиц. Различные возможности показаны на приведенном ниже рисунке. Самое важное, что стоит отметить, щелчок по зацикленным стрелочкам немедленно начинает синхронизацию соответствующей таблицы. Можно задать расписание синхронизации каждой из таблиц. Детально процесс описан ниже.



По завершение импорта вручную, в столбце, рядом со значком выводится отчет:



Щелчок по значку "Информация" выведет всю относящуюся ко внешнему импорту информацию TCA:



Настройка синхронизации по расписанию

Автоматическая синхронизация по расписанию для работы требует Gabriel (TYPO3 4.2 или ниже) или Планировщик (TYPO3 4.3+). Помимо обычных настроек Gabriel или Планировщика, нужно обратить внимание на некие специфические моменты для внешнего импорта.

Как видно из приведенного рисунка, информация по автоматической синхронизации отображается для каждой таблицы. Можно добавить или отредактировать расписание, щелкнув по значку "новая" или "карандаш", соответственно. При этом будет выведена форма, где можно ввести начальную дату (дата первого запуска, можно оставить пустой для незамедлительного включения) и частоту. Частоту можно ввести либо как количество секунд, либо использовать синтаксис для задач cron.

Щелчок по значку корзины отменит автоматическую синхронизацию.

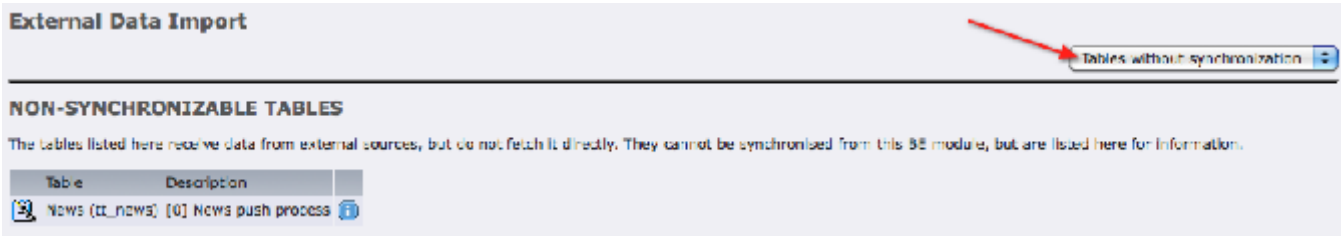
В самом низу можно настроить расписание для **всех** таблиц. Это значит, что импорт будет произведен друг за другом, в порядке приоритета.

Определения расписания недостаточно. Должна быть принята во внимание и установка верных прав пользователя. В процессе установки Gabriel создается пользователь "_cli_gabriel" (или "_cli_scheduler" для Планировщика). Этот профиль будет использоваться при синхронизации по расписанию, поэтому нужно удостовериться, что этот пользователь имеет достаточные права на совершение таких операций. В основном вот что нужно сделать:

- Разрешить для этого пользователя изменять таблицы, которые будут синхронизироваться.
- Дать пользователю доступ к странице(ам), на которых хранятся записи, то есть страницы должны находится внутри точки доступа, а сам пользователь должен иметь достаточно прав для этих страниц: "Показать страницу", "Править содержимое", "Править страницу" и "Удалить страницу" (Веб > Доступ). Конечно все это можно сделать и при помощи групп, к которой относится пользователь.

Не синхронизируемые таблицы

Вторая функция внутреннего модуля называется "Не синхронизированные таблицы", она отображает не синхронизируемые таблицы. Она крайне малоинформативна, так как для этих таблиц не могут быть предприняты никакие действия.



Разметка данных

В главе Управление ниже представлено разъяснение по разметке данных из внешних источников на существующие или вновь созданные таблицы в базе данных TYPO3. Для правильного выполнения этих операций обязательно выполнение двух условий:

- Внешние данные **должны** иметь эквивалент или первичный ключ.
- Первичный ключ **должен** хранится в каком-либо столбце базы данных TYPO3, но **не** в столбце uid, использующимся внутри TYPO3.

Первичный ключ для внешних данных используется для определения, соответствуют ли внешние данные записи уже имеющейся в базе данных TYPO3, либо необходимо создать новую запись. Записи базы данных TYPO3, не имеющие первичных ключей для внешних данных могут быть удалены при желании.

Отладка

В процессе синхронизации имеется множество потенциальных источников ошибок, начиная от неверной настройки разметки, и заканчивая ошибками прав пользователя и ошибками PHP в пользовательских функциях. При загрузки синхронизации из внутреннего модуля, создается запрос AJAX для сценария импорта. В ответ читается и отображается модуль внутреннего интерфейса.

При возникновении ошибок PHP или другой отладочной информации, происходит повреждение ответа AJAX (который ожидается в формате JSON). Для избежания этого, сценарий импорта вычищает все выводимое перед возвращением ответа. Недостаток этого метода в том, что теряется вывод ошибок и отладочной информации. Обратите внимание, что эту информацию в любом случае сложно понять. Нужно быть в состоянии прочитать сырой ответ, чтобы понять, что за ошибка или отладочная информация была в нем (например, используя комбинацию Firefox и его дополнения Firebug). *Это справедливо только для TYPO3 4.1 или 4.2. Начиная с TYPO3 4.3 выводимое всегда очищается, поэтому нет возможности отследить ошибку.*

Одним из типов вывода отладочной информации является настройка `$TYPO3_CONF_VARS[SYS][sqlDebug]`. При ее включении TYPO3 выводит полный стек отладки при ошибки запроса SQL. Эта информация удаляется, если выбран обычный ее вывод. Кроме того, можно использовать расширение, записывающее devLog (например, “devlog”) и включающее `$TYPO3_CONF_VARS[SYS][enable_DLOG]`. Таким образом, все ошибочные запросы записываются в devLog, а информация не теряется.

Как было сказано в разделе “Настройки” выше, также можно получить детализированный отчет по email. В нем будет подытожено все, что происходило в процессе синхронизации, и приведены все сообщения об ошибках, зарегистрированных в TCEmain, если таковые имелись.

Неисправности

Не выполняется автоматическая синхронизация

Может случиться, что запланированная синхронизация не происходит вовсе. Даже при включении режима отладки и просмотре devLog, не видно вызова external_import. Это может произойти при выставлении слишком большой частоты синхронизации (вроде 1 минуты, например). Если предыдущая синхронизация не завершена, Gabriel не запускает следующую. Симптомом является сообщение, вроде “[gabriel]: Event is already running and multiple executions are not allowed, skipping! CRID: xyz, UID: nn” в системном журнале (Сервис управления > Журнал). При этом нужно удалить существующее расписание и создать новое.

Синхронизация, запущенная вручную не заканчивается

Вполне может произойти, что в процессе ручной синхронизации не будет получено никакого отчета, и стрелки будут вращаться бесконечно. Это происходит при комплексной ошибке синхронизации и внутренний модуль не получает ответа. Смотрите рекомендации в "Отладка" выше.

Все существующие данные были удалены

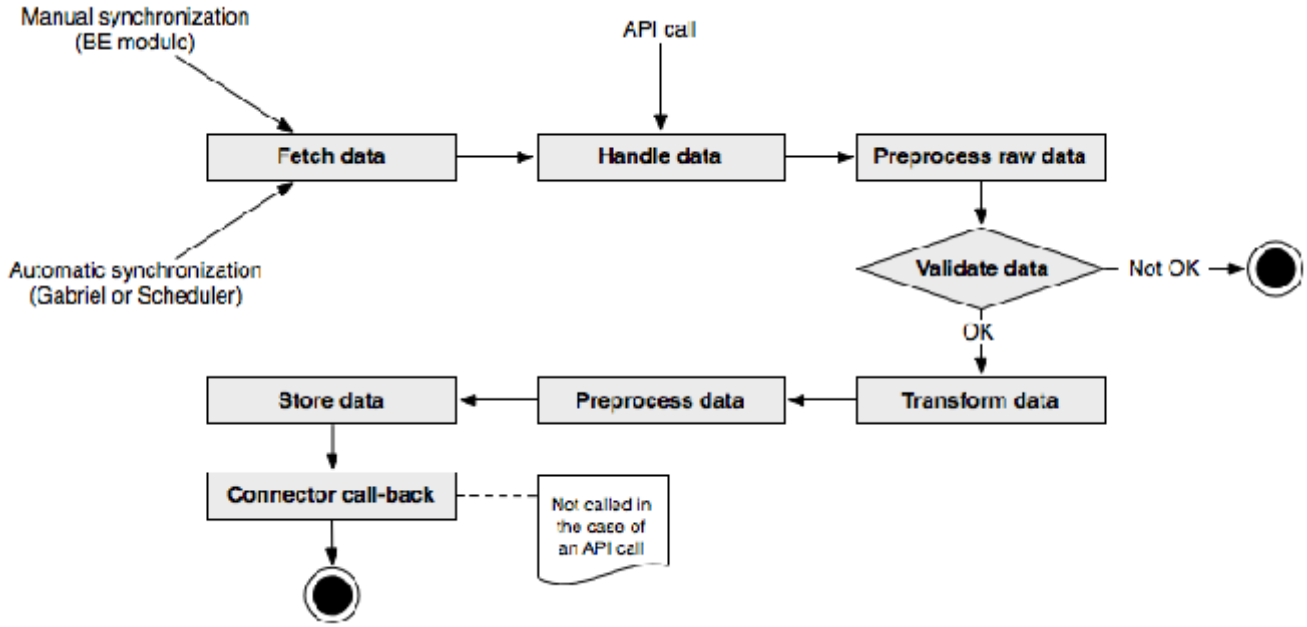
Скорее всего, внешние данные не могут быть найдены, в результате будет импортировано ноль элементов. Если

операция удаления не отключена, Внешний импорт воспримет это как сигнал к удалению всех существующих данных, так как внешний источник ничего не может предоставить.

Существует множество способов уберечь себя от этого. Прежде всего, можно отключить операцию удаления, так что никакие данные нельзя будет удалить. Если это не подходит, можно использовать параметр “minimumRecords” (смотрите "Общая настройка TCA"). Например, если всегда ожидается около 100 элементов для импорта, установите этот параметр на 100. Если в импорте присутствует число элементов меньше указанного, процесс импорта будет прерван и ничего не будет удалено.

Обзор процесса

Приведенная схема описывает процесс внешнего импорта:



Внешний импорт начинается с операции синхронизации (извлечение), сначала данные собираются из внешнего источника (при возникновении каких-либо проблем на этом этапе, процесс импорта прекращается). При использовании API этого не происходит, так как данные вводятся в процесс импорта. Следующий шаг — это "обработка данных". Именно здесь данные сохраняются во внутренних таблицах и подвергаются фильтрации. После этого данные доступны внутри процесса в виде ассоциативного массива PHP с ключами, соответствующими названиям полей базы данных, в которой они будут храниться.

Предварительная обработка сырых данных — это просто возможность вызова ловушки (hook). Затем данные проверяются. В основном проверяется, имеется ли минимальное количество данных, полученных из внешнего источника. Для дополнительных проверок доступна ловушка (hook). Если проверка не пройдена (включая базовую) процесс импорта обрывается.

Этап преобразования включает две важные операции:

1. Обработка всех простых (то есть не MM) связей (или отдельных значений).
2. Вызов заявленных пользователем функций.

На предварительной обработке ничего не делается, здесь имеется лишь ловушка (hook) для возможности работы с полным набором импортированных данных.

И наконец, данные сохраняются в базе данных. Перед этим обрабатываются MM-отношения, а перед каждым типом операций (вставка, обновление или удаление) доступна своя ловушка (hook).

На последнем этапе снова вызывается соединение, чтобы можно было выполнить необходимые операции очистки источника данных (например, отметить импортированные данные). Вызывается метод postProcessOperations() API соединения. По большей части — это место для ловушек (hook), вроде операций постобработки. Обратите внимание, что этот шаг не выполняется, если внешний импорт был начат с вызова через ее API, так как в этом случае соединения не участвуют в операции.

Учебник

В расширении “externalimport_tut” даются расширенные примеры внешнего импорта. Используются все возможные варианты настроек. Все примеры освещены в руководстве по расширению.

Управление

Для того, чтобы начать вставлять данные из внешнего источника в таблицы TYPO3, необходимо дополнить их описание TCA посредством специального синтаксиса, общей информацией в разделе “ctrl” и специфической — по каждому из столбцов. Конечно можно создать новую таблицу и поместить данные в нее.

Права пользователя

Перед тем, как забраться в дебри TCA, давайте поговорим о правах пользователя. Так как Внешний импорт опирается на TCEmain в плане хранения данных, всегда будут проверятся права пользователя при синхронизации таблиц. Но дополнительные проверки проводятся как в модуле внутреннего интерфейса, так и для автоматизированных задач, для избежания отображения важных данных или появления ненужных сообщений об ошибках.

При доступе во внутреннем модуле проверяются следующие права пользователя:

- Пользователь должен иметь по крайней мере права в списках доступа на просмотр таблицы во внутреннем модуле.
- Пользователь должен иметь права на изменение таблицы, чтобы появилась возможность ее ручной синхронизации, либо возможность запланировать ее автоматическую синхронизацию.

Здесь не проверяются точки доступа к базе данных, так что пользователь может начать синхронизацию, но по прежнему получит сообщение об ошибке, если у него нет доступа на запись на странице, где хранятся данные.

При автоматической работе синхронизации, в начале также производится проверка прав пользователя, поэтому синхронизация может быть не запущена, если пользователь CLI не имеет прав на изменение нужной таблицы. Об этом сообщается в письме с отчетом. За дополнительными сведениями о правах пользователя для автоматической синхронизации обратитесь к разделу "Настройка синхронизации по расписанию".

Общая настройка TCA

Пример синтаксиса типичного раздела “ctrl”:

```
$TCA['tx_myext_mytable'] = array (
      'ctrl' => array (
            'title'     => ...,
            ...
            'external' => array(
                  0 => array(
                        'connector' => ...,
                        'parameters' => array(
                          ...
                        ),
                        'data' => 'xml',
                        'nodetype' => 'record',
                        'reference_uid' => ...,
                        'priority' => 10,
                        'pid' => 46,
                        'enforcePid' => true
                  )
            )
      ),
);
```

Свойство “external” является индексным массивом. Доступны следующие его свойства:

Ключ	Тип данных	Описание	Область
connector	string	Подтип службы соединения. Должно быть определено только для извлекаемых данных. Если данные получаются, можно оставить пустым.	Выборка данных
parameters	array	Массив параметров, передаваемых в службу соединения. Не используется при получении данных.	Выборка данных
data	string	Формат, в котором данных возвращаются службой соединения. Может быть либо “xml”, либо “array”.	Выборка данных
nodetype	string	Название ключевых узлов в структуре XML, т.е. потомки этих узлов соответствуют данным, вставляемым в поля базы данных.	Обработка данных
reference_uid	string	Название столбца, эквивалента первичного ключа для хранилища внешних данных.	Хранение данных

Ключ	Тип данных	Описание	Область
priority	integer	Уровень приоритета для выполнения синхронизации. Некоторые таблицы могут потребовать синхронизации пред синхронизацией других для соблюдения правильной работы внешних связей. Это подсказка для пользователя и соблюдение порядка при автоматической синхронизации. Не используется при получении данных.	Отображение/Автоматизация процесса импорта
pid	integer	ID страницы, на которой должны быть сохранены импортированные данные. Можно проигнорировать, тогда будет использован pid общего хранилища (смотрите настройки).	Хранение данных
enforcePid	boolean	Если истина, то все операции, касающиеся существующих записей, будут ограничены записями, хранящимися в определенной pid (то есть, либо в свойстве выше, либо в общей настройке расширения). Это имеет два следствия: a) При проверки существования записей, записи будут выбираться только из определенной pid. b) При проверки записей для удаления, будут задействованы только записи из определенной pid. Это удобный способ защиты записей от операций, инициированных процессом импорта, чтобы они не повлияли на записи, например, созданные вручную.	Хранение данных
where_clause	string	Условия SQL, ограничивающие данные для импорта. Обновлены или удалены будут только лишь данные, соответствующие условиям. Это условие определяется выше условия "enforcePid", если оно определено. **Внимание:** это может повлечь долгое время вставки многих данных. Действительно, если некие внешние данные импортируются впервые, но не соответствуют условию "where_clause", они никогда не будут найдены для обновления. Что повлечет их многократную вставку. Всякий раз, используя свойство "where_clause", будьте готовы к неожиданно большому количеству вставок.	Хранение данных
additional_fields	string	Список полей через запятую из внешнего источника, которые должны быть доступны во время импорта, но не должны сохраняться во внутренней таблице. Обычно это поля, которые участвуют в процессе преобразования, но в конечном счете не сохраняются в базу данных.	Выборка данных
description	string	Чисто описательный текст, который должен помочь вспомнить, для чего нужна эта синхронизация. Особенно полезно, если таблица синхронизируется с несколькими источниками.	Отображение
disabledOperations	string	Список операций через запятую, которые **не** должны осуществляться. Возможные операции: insert, update и delete. Можно запретить любую из следующих операций. • **insert** (вставка) операция выполняется, когда новая запись найдена во внешнем источнике. • **update** (обновление) совершается, когда запись уже существует, но ее данные требуется обновить. • **delete** (удаление) совершается когда запись присутствует в базе данных, но отсутствует во внешнем источнике.	Хранение данных
minimumRecords	integer	Минимальное количество элементов, ожидаемых во внешнем источнике. Если элементов меньше, то импорт обрывается. Можно использовать, например, для защиты данных от удаления при ошибке во внешних данных (при этом элементы для импорта будут отсутствовать).	Проверка данных
disableLog	boolean	Установите в TRUE для отключения журналирования со стороны TCEmain. Эта настройка имеет преимущество над общей установкой "Disable logging" (смотрите "Настройки" выше).	Хранение данных

Ключ	Тип данных	Описание	Область
deleteNotSynchedRecords	boolean	*Настройка устарела. Используйте вместо нее* ***disabledOperations***. Установите в true, если записи не найденные в процессе синхронизации (то есть не существующие более в удаленном источнике) должны быть удалены. Установите в false, если они должны быть проигнорированы.	Хранение данных

Настройка столбцов

И для каждого столбца также необходим "расширенный" синтаксис для определения того, какие внешние данные в него вносятся и какая для них требуется обработка. Это также индексированный массив. Очевидно, что индексы используемые для каждого столбца должны соотносится с индексами в разделе “ctrl”. В простейшем случае — это просто ссылка на название внешних данных:

```
'field_name' => array (
      'exclude' => 0,
      'label' => '...',
      'config' => array(
            ...
      ),
      'external' => array(
            0 => array(
                  'field' => '...'
            )
      )
),
```

These are the parameters used in the column description:

Ключ	Тип данных	Описание	Область
field	string	Название или индекс поля (или узла для данных XML data), содержащего данные во внешнем источнике.	Обработка данных
attribute	string	Если тип данных XML, это свойство используется для получения значения атрибута узла (выбранного в свойстве "field" выше), а не значение самого узла.	Обработка данных
xpath	string	Свойство можно использовать для выполнения Xpath запроса относительно узла, выбранного в свойстве "field". Значение будет взято из первого узла, возвращенного запросом. Если определено свойство "attribute", оно будет применено к узлу, возвращенному запросом XPath.	Обработка данных
MM	→ MM-configuration	Определение MM-отношений, детали смотрите ниже.	Преобразование данных
mapping	→ Mapping configuration	Свойство можно использовать для связи значений из внешних данных со значениями из внутренних таблиц. Типичный пример — связь двухзначных ISO кодов стран с uid таблицы static_countries.	Преобразование данных
value	simple type	При помощи этого свойства возможно установить фиксированное значение в заданное поле. Например, это можно использовать в качестве флага для всех импортированных записей.	Преобразование данных
trim	boolean	Если установлено TRUE, каждое из значений для этого столбца будет подрезано на этапе преобразования.	Преобразование данных
rteEnabled	boolean	Если установлено TRUE, то при импорте HTML данных в поля с RTE, импортируемые данные пройдут обычный процесс преобразования RTE перед сохранением в базу данных.	Преобразование данных
userFunc	array	Это свойство может быть использовано для определения функции, вызываемой для преобразования данных каждой записи из заданного поля. Смотрите пример ниже. Обратите внимание, что userFunc вызывается **после** разметки.	Преобразование данных
excludedOperations	string	Список операций с базой данных через запятую, которые должны быть исключены для столбца. Возможные значения “insert” и “update”.	Хранение данных

Настройка разметки

Внешние значения могут быть сопоставлены со значениями из существующей таблицы TYPO3, используя свойство “mapping”.

Ключ	Тип данных	Описание	Область
table	string	Название таблицы, из которой должны читаться сопоставляемые данные.	Преобразование данных
reference_field	string	Название поля, с которым должны сравниваться внешние значения.	Преобразование данных
value_field	string	Название поля, из которого берутся размеченные значения. Если не определено, по умолчанию используется “uid”.	Преобразование данных

where_clause	string	SQL условие (без ключевого слова “WHERE”), применяемое к связанной таблице. Обычно предназначено для зеркала свойства “foreign_table_where” полей типа select.
		Но в этом случае невозможно использовать маркеры.
		Поэтому, если имеется что-то типа:
		
		`'foreign_table_where' => 'AND pid = ###PAGE_TSCONFIG_ID###'`
		
		в TCA для столбца, необходимо заменить маркер на жестко закодированное значение, например
		
		`'where_clause' => 'pid = 42'`
		
		Обратите внимание, что условие не должно начинаться с ключевого слова “AND”.
			Преобразование данных

valueMap	array	Фиксированная хэш таблица для разметки. Вместо использования таблицы базы данных для соответствия внешних значений внутренним, это свойство делает возможным использование простого списка пара ключ-значение. Ключи соответствуют внешним значениям.	Преобразование данных
mapping_method	string	Значение может быть "strpos" или "stripos". Обычно соответствие значений подразумевает строгое равенство. Это свойство может использоваться для "смягчения" соответствия. Соответствие будет признано, если внешнее значение входит в значения, на которые указано в поле "reference_field". "strpos" означает соответствие чувствительное к регистру, в то время как "stripos" — не чувствительное. Проявляйте осторожность, используя это свойство. Так как соответствие не строгое, то оно может быть и ошибочным. Необходимо просмотреть данные после подобного импорта.	Преобразование данных
mapping_symmetric	boolean	Это свойство дополняет "mapping_method" выше. Если true, то в процессе импорта не только оценивается вхождение внешнего значения в размеченные, но и обратное, т.е. разметка значений внутри внешнего значения.	Преобразование данных

Пример настройки TCA. Определено свойство “value_field”, хотя в этом случае это и не обязательно, так как его значение – “uid”, которое используется по умолчанию.

```
'field_name' => array (
      'exclude' => 0,
      'label' => '...',
      'config' => array(
            ...
      ),
      'external' => array(
            0 => array(
                  'field' => '...',
                  'mapping' => array(
                        'table' => name of foreign table,
                        'reference_field' => foreign MM key,
                        'value_field' => 'uid'
                  )
            )
      )
),
```

Соображение о "мягкой разметке"

Важно понять, как свойство "mapping_method" влияет на процесс соответствия. Соответственно, попытаемся разметить наобум введенные названия стран в таблицу `static_countries` внутри TYPO3. Это может оказаться не столь просто, в зависимости от того, как названия были введены во внешний источник. Например, "Australia" не строго соответствует официальному названию "Commonwealth of Australia". Но настроив "mapping_method" на "strpos", получим соответствие, так как "Australia" входит в "Commonwealth of Australia"

Настройка пользовательских функций

Пример настройки вызова пользовательской функции.

```
'field_name' => array (
      'exclude' => 0,
      'label' => '...',
      'config' => array(
            ...
      ),
      'external' => array(
            0 => array(
                  'field' => '...',
                  'userFunc' => array(
                        'class' =>
'EXT:external_import/samples/class.tx_externalimport_transformations
.php:&tx_externalimport_transformations',
                        'method' => 'parseDate',
                        'params' => array(
                          'function' => 'date',
                          'format' => 'd.m.Y'
                        )
                  )
            )
      )
),
```

Пользовательская функция требует трех параметров. Первый ("class") – название создаваемого класса. Оно должно даваться с указанием пути, при этом файл будет автоматически включен. Обратите внимание на "&" перед названием класса. Это позволяет создать единственный экземпляр. Следующий параметр ("method") определяет вызываемый метод. Третий параметра ("params") не обязателен. Это массив, который может содержать любое количество данных. Он передается в метод.

В приведенном выше примере мы используем образец класса, поставляемого вместе с расширением external import, который можно использовать для анализа даты, и возвращения ее либо в виде timestamp (время в определенном шаблоне), либо отформатировать используя одну из функций PHP `date()` или `strftime()`.

За деталями по созданию пользовательских функций обратитесь к Руководству разработчиков ниже.

Настройка MM-отношений

Сложнее перенастроить MM-отношения после импорта:

```
'external' => array(
      0 => array(
            'MM' => array(
                  'mapping' => array(
                        'table' => name of foreign table,
                        'reference_field' => foreign MM key,
                  ),
                  'additional_fields' => array(
                        TYPO3 field name => external data field name
                  ),
                  'sorting' => 'field',
            )
      )
)
```

Параметры, используемые в настройках MM:

Ключ	Тип данных	Описание	Область
mappings	array	*Свойство устарело. Смотрите "mapping" ниже.*	Хранение данных
mapping	→ Mapping configuration	Схоже у уже описанным выше свойством "mapping". Используется для определения связанной таблицы и столбца, содержащего внешний первичный ключ.	Хранение данных

Ключ	Тип данных	Описание	Область
additional_fields	array	Список полей, которые должны сохранятся для локальных и внешних ключей в таблице MM. Для каждого такого поля нужно определить соответствие полю таблицы MM TYPO3 поле внешних данных.	Хранение данных
multiple	boolean	Если некоторые связи mm многократно существуют во внешних данных (из-за разных дополнительных полей), нужно установить это свойство в 1, и они будут сохранены (иначе TCEmain будет брать во внимание только уникальные паоы uid_local, uid_foreign).	Хранение данных
sorting	string	Говорит о том, что данные должны быть отсортированы по этому полю внешних данных. Обратите внимание, что внешний импорт в хранении данных полагается на TCEmain, который устанавливает собственную нумерацию для сортировки, и поэтому данное значение никогда не используется в сортировке, но только для упорядочивания данных. Поэтому, если записи во внешнем источнике уже отсортированы, нет необходимости указывать свойство “sorting”.	Хранение данных

Замечание: при использовании свойств “additional_fields” и/или “multiple” выполняются дополнительные операции с базой данных, так как это не обычное поведение для TYPO3 MM-отношений. Это должно быть возможно с IRRE, но пока не поддерживается.

Руководство для разработчиков

API внешнего импорта

Функционал внешнего импорта легко задействовать. Необходимо лишь подготовить данные в понятном формате (структура XML или recordset) и вызвать соответствующий метод. Необходимо включить class.tx_externalimport_importer.php и сделать следующий вызов:

```
$importer = t3lib_div::makeInstance('tx_externalimport_importer');
$importer->importData($table, $index, $rawData);
```

Параметры вызова:

Название	Тип	Описание
$table	string	Название таблицы для хранения данных.
$index	integer	Индекс соответствующих внешних настроек.
$rawData	mixed	Данные для сохранения, XML или recordset.

Это особенно полезно в сочетании с расширением Remote Server extension (ключ: remote_server). При этом можно вызывать внутренний интерфейс TYPO3 и переслать в него данные, обработать их и сохранить в локальной таблице, используя API внешнего импорта.

Пользовательские функции

Расширение внешнего импорта может вызывать пользовательские функции для любого поля, в которое импортируются внешние данные. Образец функции находится в `samples/class.tx_externalimport_transformations.php`. В основном функция получает три параметра:

Название	Тип	Описание
$record	array	Полная обрабатываемая запись. Это дает возможность обратиться к другим полям той же записи при необходимости в процессе преобразования .
$index	string	Ключ преобразуемого поля. Изменение других значений в записи невозможно, так как запись передается по значению а не по ссылке. Могут быть преобразованы и возвращены только поля, соответствующие этому ключу.
$params	array	Дополнительные параметры, передаваемые в функцию. Это спецефично для каждой из функций, и может быть даже не указано. Если свойство "params" не определено, внешний импорт передаст в функцию пустой массив.

Функция должна возвращать только значение преобразованного поля.

Ловушки (hooks)

Процесс внешнего импорта содержит следующие ловушки (hooks):

- **preprocessRawRecordset**: эта ловушка делает возможным управление данными сразу после их получения из удаленного источника, но уже после преобразования их в массив PHP, независимо от их оригинального формата. Ловушка получает в качестве параметров полный набор данных и обратную ссылку на вызывающий объект (экземпляр класса tx_externalimport_importer). Ожидается возвращение также полного набора данных.
- **validateRawRecordset**: ловушка вызывается на этапе проверки данных. Используется для проверки почти необработанных данных (прошедших лишь "preprocessRawRecordset") и решения, нужно ли продолжать импорт. Ловушка получает в качестве параметров полный набор данных и обратную ссылку на вызывающий объект (экземпляр класса tx_externalimport_importer). Ожидается возвращения истины, если импорт может быть продолжен, или лжи, если импорт нужно остановить.
  Обратите внимание на следующее: если условие минимального количества импортируемых данных не соблюдено, ловушка не вызывается совсем. Импорт обрывается до этого. Если в ловушке зарегистрировано несколько методов, то первый из возвращающих ложь прерывает импорт. Дальнейшие методы не вызываются.
- **postprocessRecordset**: схоже с "preprocessRawRecordset", но выполняется после этапа преобразования, то есть прямо перед сохранением в базу данных. Ловушка получает в качестве параметров полный набор данных и обратную ссылку на вызывающий объект (экземпляр класса tx_externalimport_importer). Ожидается возвращение также полного набора данных.

- **updatePreProcess**: ловушку можно использовать для изменения записи прямо перед ее обновлением в базе данных. Ловушка вызывается для каждой обновляемой записи. Ловушка получает в качестве параметров полный набор данных и обратную ссылку на вызывающий объект (экземпляр класса tx_externalimport_importer). Ожидается возвращение также полного набора данных.
- **insertPreProcess**: схоже с "updatePreProcess", но для операции вставки (insert).
- **deletePreProcess**: ловушка используется для изменения списка записей, которые должны быть удалены. В качестве первого параметра получает список первичных ключей, соответствующих удаляемому набору данных. Второй параметр — ссылка на вызывающий объект (опять же, экземпляр класса tx_externalimport_importer). Ожидается, что метод вызывается для возвращения списка первичных ключей.
- **datamapPostProcess**: эта ловушка вызывается после обновления или вставки всех данных, используя TCEmain. Может быть использован для любых последующих операций. В качестве параметров получается название изменяемой таблицы, список uid записей в ней (включая uid новых записей) и обратную ссылку на вызываемый объект (экземпляр класса tx_externalimport_importer). Каждая запись содержит дополнительное поле "tx_exteralimport:status", содержащее либо "insert", либо "update", в зависимости от совершаемой над записью операции.
- **cmdmapPostProcess:** эта ловушка вызывается после удаления всех данных, используя TCEmain. В качестве параметров получается название таблицы, список uid удаляемых в ней записей и обратную ссылку на вызываемый объект (экземпляр класса tx_externalimport_importer).

Список задач

На Forge имеется карта развития разработки этого расширения:

http://forge.typo3.org/projects/roadmap/extension-external_import

Ниже приведены некоторые идеи, приоритетные на данный момент:

- обработка локализуемых записей;
- обработка ссылающихся на себя таблиц при вставки новых записей;
- Поиск в IRRE для обработки MM-отношений, использующих дополнительные поля или многократно повторяющихся.

Список изменений

Версия	Изменения:
1.2.0	Added possibility to retrieve value from attributes and to use XPath in XML data Added configuration for importing rich-text fields Exceptions thrown by connectors now interrupt import process Added softer matching methods to mapping process Added SQL condition to match records considered for import process Added trim property on import data Added flag to disable TCEmain logging
1.1.0	Added support for additional where clause in foreign mappings Added connector call-back as a post-processing step
1.0.0	Added early check of user rights in automated synchronisation Added check of user rights in BE module display Added option for limiting preview/debug output size Added TCA property to exclude some fields from insert or update operations Added display of TCA external configuration in BE module
0.11.2	Fixed abusive display of validation error message
0.11.0 – 0.11.1	Added reporting by email Added automatic synchronisation per import configuration Made it possible to delete a defined automatic synchronisation Added support for the Scheduler (TYPO3 4.3+) Added a process to abort the import
0.10.0	Added fixed values and fixed value maps
0.9.0	Added "clear output" option
0.8.1	Added preprocessing hook on raw recordset
0.8.0	Introduced possibility to synchronise a table with multiple external sources Added support for user functions for transforming external data before storage Introduced API for pushing data into the external import Added hooks for preprocessing records before insert, update and delete Added property for limiting records manipulation to a given pid Added property for limiting allowed operations Cleaned up MM-mapping syntax Corrected bugs in mapping feature Updated manual with instructions for new features and some troubleshooting help Raised status to beta
0.7.0 – 0.7.x	Internal releases
0.6.2	Updated manual with missing notes about array data format
0.6.1	Added array data format handling Added user rights setup instructions in the manual
0.6.0	Introduced use of TCEmain for proper data manipulation Cleaned up extended TCA syntax Added hook for pre-processing data before storage

Версия	Изменения:
0.5.0	First public release