На платформе Alfresco ECM

Импорт и экспорт в Citeck EcoS

2015

Оглавление

Введение

Когда используется импорт и экспорт

Чаще всего ECM система внедряется в существующую инфраструктуру организации, и данные уже где-то хранятся. Поэтому актуальными являются задачи импорта данных из внешних источников и экспорта данных во внешние приемники.

Задачи импорта и экспорта могут быть разовыми и постоянными. Разовые задачи импорта обычно требуются в начале жизненного цикла системы для импорта необходимых данных из других систем. Разовые задачи экспорта обычно требуются при появлении новых систем, которым необходимы данные, хранящиеся в системе, или в конце жизненного цикла системы, когда производится миграция на заменяющую систему.

Постоянные задачи импорта и экспорта обычно требуются для штатного взаимодействия системы с ее окружением. Например, можно постоянно импортировать справочник из доступной БД или загружать входящие файлы из папки. Примером постоянного экспорта является экспорт в БД для составления отчетности. Система Citeck EcoS содержит в себе сервис синхронизации, предназначенный для постоянного осуществления импорта и экспорта.

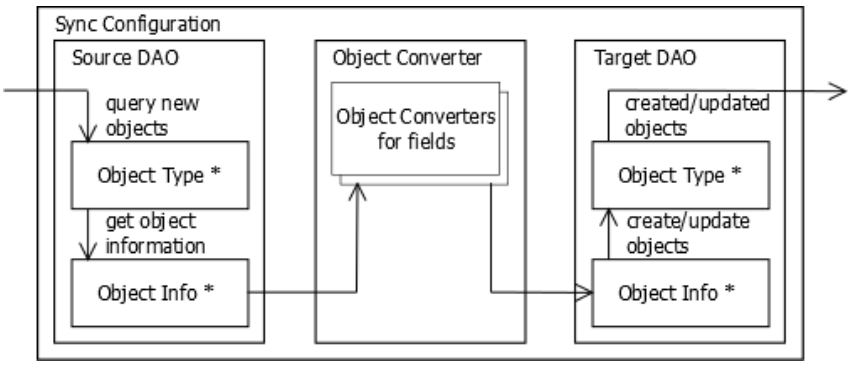
Архитектура сервиса синхронизации

Сервис синхронизации оперирует конфигурациями синхронизации. Каждая конфигурация – это набор из трех компонентов:

- Источник данных (Source DAO)
- Конвертер данных (Converter)
- Приемник данных (Target DAO)

Синхронизация с произвольной конфигурацией происходит по следующей схеме:

1. Source DAO получает очередной набор объектов для синхронизации (либо все, либо только обновленные с момента последней синхронизации);
2. Source DAO преобразует полученные объекты в формат Object Info;
3. Object Converter обеспечивает согласование форматов Object Info между Source DAO и Target DAO; для преобразования отдельных полей могут использоваться дополнительные объекты Object Converter;
4. Target DAO создает или обновляет объекты в соответствии с полученной информацией.



Описание возможностей синхронизации

В Citeck EcoS есть несколько доступных реализаций источников и приемников данных:

- репозиторий Alfresco (источник и приемник);
- JDBC-совместимые базы данных (источник и приемник);
- папки XML-файлов (источник);
- папки произвольных файлов (источник);
- база данных отчетности (приемник).

Также доступны некоторые реализации конвертеров:

- константное значение
- замена строки по регулярному выражению
- преобразование карты в строку по шаблону
- преобразование карты в объект репозитория
- преобразование внешнего ключа в объект репозитория
- преобразование объекта репозитория в карту
- преобразование карты в другую карту
- преобразование строки в карту с единственным ключом
- преобразование строки в дату
- текущая дата
- вызов цепочки конвертеров

Примеры конфигураций синхронизации

Простейшая конфигурации синхронизации

Конфигурация синхронизации выполняется в context-файле Spring (файлы *-context.xml). В следующем примере сконфигурирован импорт справочника из таблицы внешней БД в Alfresco. Каждый объект конфигурации (источник, конвертер, приемник и собственно конфигурация) задаются в XML-нотации Spring Context.

Начнем с источника данных. Прежде всего должен быть сконфигурирован обычный JDBC Data Source, в нем указываются параметры подключения к базе данных:

```
<bean id="test-import-datasource"
      class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="org.postgresql.Driver"/>
    <property name="url" value="jdbc:postgresql://localhost:5432/testdata"/>
    <property name="username" value="alfresco"/>
    <property name="password" value="password"/>
</bean>
```

Далее конфигурируется собственно источник данных JDBC:

```
<bean id="test-import-source" parent="ExternalSync.DAO.JDBC">
    <property name="dataSource" ref="test-import-datasource" />
    <property name="queryConstructor">
        <bean parent="ExternalSync.QueryConstructor.SQL">
            <property name="allQuery">
                <value>SELECT id, name, text FROM data</value>
            </property>
            <property name="newQuery">
                <value>
                    SELECT id, name, text FROM data WHERE version > :version
                </value>
            </property>
            <property name="currentVersionQuery">
                <value>SELECT MAX(version) AS version</value>
            </property>
        </bean>
    </property>
</bean>
```

Как видно из конфигурации источник данных ссылается на определенный ранее JDBC DataSource и задает несколько запросов:

- `allQuery` – запрашивает все записи из источника данных;
- `newQuery` – запрашивает новые записи (с версией больше, чем заданная);
- `currentVersionQuery` – запрашивает текущую версию данных.

Запросы `newQuery` и `currentVersionQuery` необходимы для дифференциальной синхронизации. Это работает следующим образом: после каждой синхронизации система запоминает текущую версию записей (`currentVersionQuery`), а при следующей дифференциальной синхронизации запрашивает записи, которые изменились, начиная с этой версии (`newQuery`).

Сконфигурируем приемник данных:

```
<bean id="test-import-target" parent="ExternalSync.DAO.Node">
    <property name="nodeType"
        value="{http://www.citeck.ru/model/test/1.0}testRecord" />
    <property name="newNodesRootPath"
        value="/app:company_home/st:sites/cm:test/cm:documentLibrary" />
    <property name="keys">
        <set><value>extsync:id</value></set>
    </property>
</bean>
```

Как видно из конфигурации, в источнике данных указывается тип данных репозитория, который соответствует записям во внешней таблице, а также путь, где будут создаваться новые объекты, и набор ключевых свойств. Все вместе это работает следующим образом: когда из внешнего источника приходят данные, которые нужно сохранить в репозитории, мы еще не знаем, существуют они в репозитории или нет. Для этого делается Lucene-запрос следующего вида:

```
TYPE:"{http://www.citeck.ru/model/test/1.0}testRecord" AND @extsync\:id:"45"
```

Другими словами, ищутся объекты по ключу: если что-то найдено, то это необходимо обновить, а если нет – то нужно создать новый объект. Новые объекты создаются по пути `newNodesRootPath` и с дочерней ассоциацией `newNodesAssocType` (по умолчанию `cm:contains`).

В этом примере источник данных возвращает карты (maps) следующего вида:

```
{
    "id": 1,
    "name": "Sergey",
    "text": "Hi, I am Sergey"
}
```

и их необходимо преобразовать к виду, подходящему для поиска и создания (обновления) объектов репозитория:

```
{
    "extsync:id": 1,
    "test:author": "Sergey",
    "test:theme": "Hi, I am Sergey"
}
```

Эту задачу решает конвертер. Как видно, в данном случае необходимо просто заменить названия ключей карты названиями свойств репозитория. Для этого достаточно определить следующий конвертер:

```
<bean id="test-import-converter"
      parent="ExternalSync.Converter.ImportRecord">
    <property name="propertyMapping">
        <map>
            <entry key="extsync:id"
                   value="id" />
            <entry key="{http://www.citeck.ru/model/test/1.0}author"
                   value="name" />
            <entry key="{http://www.citeck.ru/model/test/1.0}theme"
                   value="text" />
        </map>
    </property>
</bean>
```

Свойство `propertyMapping` конвертера определяет отображение свойств репозитория в названия ключей входящей карты. В этом отображении в качестве ключа выступает свойство репозитория, а в качестве значения – ключ входящей карты. Таким образом, можно несколько свойств репозитория завязать на один и тот же ключ входящей карты.

Если помимо преобразования ключей требуется преобразование значений, то для этого используется свойство конвертера `propertyConverters`, которое определяет как конвертировать значение для каждого из свойств репозитория. Например, следующая конфигурация убирает пробелы в начале и конце значения:

```
...
  <property name="propertyConverters">
    <map><entry key="test:theme" value-ref="test-spaces-conv" /></map>
  </property>
...
<bean id="test-spaces-conv" parent="ExternalSync.Converter.RegexpReplace">
    <property name="regexp" value="^ +| +$" />
    <property name="replace" value="" />
</bean>
```

Когда заданы источник, приемник и конвертер, можно определить конфигурацию следующим образом:

```
<bean id="test-import-config" parent="ExternalSync.Config">
    <property name="name" value="test-import" />
    <property name="sourceDAO" ref="test-import-source" />
    <property name="targetDAO" ref="test-import-target" />
    <property name="converter" ref="test-import-converter" />
</bean>
```

Помимо источника, приемника и конвертера в конфигурации также указывается ее имя. При запуске синхронизации указывается это имя и режим синхронизации (полный или дифференциальный).

Варианты запуска синхронизации

Для запуска синхронизации есть следующие варианты:

- Запуск по запросу
- Запуск по расписанию

В любом варианте перед запуском синхронизации актуальная конфигурация синхронизации должна быть записана в файле *-context.xml и сервер должен быть перезагружен.

Для запуска по запросу необходимо отправить на сервер следующий HTTP-запрос:

```
http://localhost:8080/alfresco/service/citeck/ext-sync/test-import?mode=full
```

Здесь `http` – протокол, `localhost` – имя хоста Alfresco, `8080` – порт Alfresco, `alfresco` – имя веб-приложения репозитория Alfresco, `service` – имя сервлета веб-скриптов, `citeck/ext-sync/{sync-name}` – адрес веб-скрипта запуска синхронизации, `test-import` – название конфигурации синхронизации, `full` – режим синхронизации (полный). В этом запросе можно изменить название конфигурации и режим (`full` или `diff`).

При запуске синхронизации в лог-файле отражается ход синхронизации, следующим образом:

```
test-import: starting full synchronization
test-import: current version: {version=1}
test-import: received 4 objects to update
test-import: synchronization finished with success, created: 0, updated: 4,
deleted: 0 objects
```

Для запуска синхронизации по расписанию следует добавить в Spring-контекст специальный bean. В следующем примере указано, что конфигурация `test-import` будет отправляться на полную синхронизацию каждый час (выражение cron: `0 0 * * * ?` – это маска – секунды минуты часы дни_месяца месяцы дни_недели).

```
<bean id="test-import-job" class="org.alfresco.util.CronTriggerBean">
    <property name="jobDetail">
        <bean class="org.springframework.scheduling.quartz.JobDetailBean">
            <property name="jobClass">
                <value>ru.citeck.alfresco.acm.sync.ExternalSyncJob</value>
            </property>
            <property name="jobDataAsMap">
                <map>
                    <entry key="ExternalSyncService"
                           value-ref="ExternalSyncService" />
                    <entry key="SyncConfigName" value="test-import" />
                    <entry key="FullSync" value="true" />
                </map>
            </property>
        </bean>
    </property>
    <property name="enabled" value="true" />
    <property name="scheduler" ref="schedulerFactory" />
    <property name="cronExpression" value="0 0 * * * ?" />
</bean>
```

Синхронизация связанных справочников

В предыдущем разделе свойство test:author заполнялось строкой из атрибута name внешней таблицы. Что если test:author – это не свойство, а ассоциация, а строка – это некий внешний ключ, например, имя пользователя. В таком случае при синхронизации вместо установки свойства система должна по внешнему ключу найти (или создать) в репозитории связанный объект и создать ассоциацию на основной объект.

Это можно сделать с помощью конвертера `ExternalSync.Converter.Assoc`. Этот конвертер позволяет сконвертировать произвольное значение в объект репозитория. Для начала требуется определить для связанных объектов отдельный приемник (Target DAO):

```
<bean id="test-import-people" parent="ExternalSync.DAO.Node">
    <property name="nodeType"
        value="cm:person" />
    <property name="keys">
        <set><value>cm:userName</value></set>
    </property>
</bean>
```

Этот приемник связан с типом пользователь (cm:person), а в качестве ключа используется имя пользователя (cm:userName). Этого достаточно для того, чтобы сконфигурировать конвертер:

```
<bean id="test-import-people-converter"
      parent="ExternalSync.Converter.Assoc">
    <property name="targetDAO" ref="test-import-people" />
    <property name="converter">
        <bean parent="ExternalSync.Converter.ImportRecord">
            <property name="propertyMapping">
                <map><entry key="cm:userName" value="username" /></map>
            </property>
        </bean>
    </property>
</bean>
```

Конфигурация конвертера `ExternalSync.Converter.Assoc` похожа на конфигурацию синхронизации: в ней указаны приемник (`targetDAO`) и конвертер (`converter`). При этом источник не требуется, поскольку при конвертации данные уже получены из источника (в данном случае требуется сконвертировать внешний ключ в объект репозитория). Конвертер работает следующим образом:

- полученные данные отправляются во вложенный конвертер, задача которого – привести данные к виду, требуемому приемнику данных;
- когда данные сконвертированы, приемник опрашивается – существуют ли у него уже подходящие объекты; возможны три варианта ответа:
    - если подходящий объект один, то он возвращается;
    - если подходящих объектов нет, то по умолчанию ничего не происходит (конвертер возвращает пустой массив или null);
    - если подходящих объектов несколько, то по умолчанию возвращается первый.

Поведение для второго и третьего варианта, можно переопределить с помощью свойств (соответственно) `zeroStrategy` и `manyStrategy`, указав им одну из следующих стратегий:

- PROCESS_ALL – отработать все полученные результаты;
- PROCESS_FIRST – отработать первый полученный результат;
- CREATE_NEW – создать новый объект;
- MERGE – объединить все полученные результаты в один (пока не реализовано);
- FAIL – выдать сообщение об ошибке.

Сконфигурированный конвертер `test-import-people-converter` ожидает на вход карту с ключом username. Его можно использовать в составе ранее определенного конвертера `test-import-converter` для того, чтобы организовать ассоциацию:

```
...
<property name="associationMapping">
    <map>
        <entry key="{http://www.citeck.ru/model/test/1.0}author"
               value="name" />
    </map>
</property>
<property name="associationConverters">
    <map>
        <entry key="{http://www.citeck.ru/model/test/1.0}author">
            <bean parent="ExternalSync.Converter.Sequence">
                <property name="converters">
                    <list>
                        <bean parent="ExternalSync.Converter.MakeMap">
                            <property name="key" value="username" />
                        </bean>
                        <ref bean="test-import-people-converter" />
                    </list>
                </property>
            </bean>
        </entry>
    </map>
</property>
```

Импорт файлов из папки

Для импорта файлов из папки используется источник ExternalSync.DAO.File. Для этого источника можно указать папку, из которой будут импортироваться файлы, а также (опционально) папку, в которую будут перемещаться успешно импортированные файлы, и папку, в которую будут перемещаться файлы, которые не удалось импортировать.

```
<bean id="test-import-file-source" parent="ExternalSync.DAO.File">
    <property name="sourceDir" value="/mnt/import-files/source"/>
    <property name="errorDir" value="/mnt/import-files/errors"/>
    <property name="successDir" value="/mnt/import-files/archive"/>
</bean>
```

Каждый объект этого источника данных представляет собой карту с двумя ключами:

- `contentFile` – содержимое файла
- `contentFileName` – имя файла

В простейшем случае можно сделать `propertyMapping`, который отображает эти ключи в соответствующие свойства Alfresco:

```
<property name="propertyMapping">
    <map>
        <entry key="cm:name" value="contentFileName" />
        <entry key="cm:content" value="contentFile"/>
    </map>
</property>
```

И далее использовать этот источник по аналогии с примерами выше.

В более сложном случае, можно выделять из имени файла отдельные составляющие (с помощью RegexpReplace-конвертера) и далее класть их в свойства или ассоциации (с помощью Assoc-конвертера).

Для дифференциальной синхронизации источник файлов предоставляет несколько вариантов.

- Использовать одну папку, в качестве версии он запоминает время изменения файла (атрибут файла в файловой системе), и при дифференциальной синхронизации запрашиваются файлы, которые изменились позже, чем последнее время. Этот вариант подходит, если количество файлов небольшое.
- Использовать папки `successDir` и `errorDir`. После успешного импорта файл перемещается в папку `successDir` (если она указана). Если же импорт не удался, то файл перемещается в папку `errorDir` (если она указана) вместе с файлом описания ошибки (файл с расширением `.error`). После исправления ошибки в файле его можно вернуть в папку `sourceDir` и импорт будет повторен. В этом варианте необходимо использовать полную синхронизацию, т.е. читать все файлы из папки.

Импорт XML-файлов

Для импорта XML-файлов в EcoS реализован специальный источник данных. По аналогии с источником файлов (см. предыдущий раздел), у него можно указать свойства sourceDir, successDir, errorDir, а также набор свойств, задающих структуру XML-файла:

- `objectPath` – XPath-выражение, задающее расположение XML-элементов для импорта (относительно корневого XML-элемента);
- `attributesMap` – карта, отображающая ключ выходной карты в XPath-выражение, из которого можно получить значение для этого ключа (относительно импортируемого XML-элемента);
- `defaultNamespace` – пространство имен XML по умолчанию – используется для файла, если в нем не указано иное;
- `xpathNamespaces` – карта с дополнительными пространствами имен XML – отображает префикс в пространство имен.

Допустим входные XML-файлы имеют следующий формат:

```
<?xml version="1.0" encoding="UTF-8"?>
<documents>
    <document id="1">
        <author>Sergey</author>
        <text>Hi, I am Sergey</text>
    </document>
    <document ...>
        ...
    </document>
    ...
</documents>
```

Тогда конфигурацию источника данных XML можно записать следующим образом:

```
<bean id="test-import-xml-source" parent="ExternalSync.DAO.XML">
    <property name="sourceDir" value="/mnt/import-xml/source"/>
    <property name="errorDir" value="/mnt/import-xml/errors"/>
    <property name="successDir" value="/mnt/import-xml/archive"/>
    <property name="objectPath" value="/def:documents/def:document"/>
    <property name="attributesMap">
        <map>
            <entry key="id"   value="./@id"/>
            <entry key="name" value="./def:name/text()"/>
            <entry key="text" value="./def:text/text()"/>
        </map>
    </property>
    <property name="defaultNamespace"
              value="http://www.citeck.ru/schema/test/1.0" />
    <property name="xpathNamespaces">
        <map>
            <entry key="def" value="http://www.citeck.ru/schema/test/1.0" />
        </map>
    </property>
</bean>
```

Сконфигурированный таким образом источник данных возвращает карты с ключами id, name, text, т.е. то же самое, что JDBC-источник в первом примере, поэтому можно определить конфигурацию импорта, использующую тот же конвертер и приемник, что и в первом примере:

```
<bean id="test-import-xml-config" parent="ExternalSync.Config">
    <property name="name" value="test-import-xml" />
    <property name="sourceDAO" ref="test-import-xml-source" />
    <property name="targetDAO" ref="test-import-target" />
    <property name="converter" ref="test-import-converter" />
</bean>
```

Каталог компонентов синхронизации

Доступные компоненты синхронизации (источники, приемники, конвертеры) задокументированы в виде абстрактных или конкретных `bean`-ов в файле `sync-services-context.xml`. В каждом из них указаны некоторые значения по умолчанию, а также перечислено, какие свойства должны быть обязательно заданы, а какие могут быть пропущены.

Если `bean` объявлен абстрактным (`abstract="true"`), то для его использования необходимо создать дочерний `bean`:

```
<bean id="..." parent="abstract-bean-id">
    <property name="..." value="..." />
    ...
</bean>
```

Если `bean` не объявлен абстрактным, то он конкретный и для его использования можно как создать дочерний `bean` (см. выше), так и использовать его непосредственно, например:

```
<property name="..." ref="concrete-bean-id" />
<property name="...">
    <ref bean="concrete-bean-id" />
</property>
<property name="...">
    <map>
        <entry key="..." value-ref="concrete-bean-id" />
    </map>
</property>
```

Экспорт данных для отчетности

В системе EcoS настроены предустановленные конфигурации для экспорта данных во внешнюю базу данных отчетности:

- `reporting-export-nodes` – экспортирует объекты типа `cm:cmobject` (и всех подтипов, включая `cm:content`, `cm:folder`, `dl:dataListItem` и дочерних типов) в таблицу `nodes`;
- `reporting-export-people` – экспортирует объекты типа cm:person (пользователи) в таблицу `people`.

При настройках доступа к БД по умолчанию экспорт происходит в базу данных alfrescoreporting на той же СУБД, с тем же логином и паролем, что и к основной БД. Для включения экспорта по расписанию достаточно в конфигурационном файле `alfresco-global.properties` указать:

```
reporting.sync-nodes.enabled=true
reporting.sync-people.enabled=true
```

По умолчанию производится дифференциальная синхронизация раз в полчаса, расписание также можно изменить в файле `alfresco-global.properties`:

```
reporting.sync-nodes.cronExpression=0 0/30 * * * ?
reporting.sync-people.cronExpression=0 15/30 * * * ?
```

Все настройки из файла `alfresco-global.properties` применяются только после перезагрузки Alfresco.

База данных для отчетности, а также таблицы nodes и people должны быть уже созданы, и на них должны быть выданы полные права для пользователя alfresco (под которым осуществляется выгрузка). Например, для PostgreSQL это можно сделать с помощью следующих команд:

```
CREATE DATABASE alfrescoreporting;
GRANT ALL ON DATABASE alfrescoreporting TO alfresco;
\c testdata
CREATE TABLE nodes (sys_node_uuid CHAR(36) PRIMARY KEY);
CREATE TABLE people (sys_node_uuid CHAR(36) PRIMARY KEY);
```

Формат таблиц отчетности следующий: для каждого свойства заводится отдельный столбец в таблице, который называется коротким именем свойства (все недопустимые символы заменяются на символ _, например, свойство `sys:node-uuid` отображается в колонку `sys_node_uuid`). При появлении новых свойств они добавляются как дополнительные колонки в таблице. Для ассоциаций в системе присутствуют зеркалирующие свойства, например, ассоциация `letters:addressee` будет отображена в колонку `letters_addressee_added`. Наконец, в таблице есть несколько специальных колонок:

- `noderef` – содержит nodeRef объекта;
- `parent` – содержит nodeRef родительского объекта;
- `type` – содержит короткое имя типа.

Инструменты для разового импорта и экспорта

Импорт оргструктуры из XML-файла

В системе реализован веб-скрипт для простого разового импорта оргструктуры из XML-файла. Входной XML-файл должен иметь следующий формат:

```
<orgstruct>
    <record>
        <company>Рога и копыта</company>
        <branch>Филиал в г. Тула</branch>
        <department>Бухгалтерия</department>
        <employee>Бухгалтер</employee>
        <userid>kuznetsova</userid>
    </record>
    ...
</orgstruct>
```

Названия тэгов могут быть произвольными и настраиваются в конфигурационном файле `tomcat/shared/classes/alfresco/extension/templates/webscripts/ru/citeck/import/orgstruct-import.get.config.xml` (при необходимости этот файл нужно создать):

```
<config>
    <recordName>record</recordName>
    <groupTypes>
        <groupType id="company"              xmlField="company"
                   shortName="{type}{index}" displayName="{name}"
                   groupType="branch"        groupSubtype="company"
                   scope="global" />
        <groupType ... />
        ...
    </groupTypes>
    <userTypes>
        <userType id="user" xmlField="userid" alfField="cm:userName"
                  scope="employee,manager" />
    </userTypes>
</config>
```

Пример полной конфигурации импорта оргструктуры можно посмотреть в файле `tomcat/webapps/alfresco/WEB-INF/classes/alfresco/templates/webscripts/ru/citeck/import/orgstruct-import.get.config.xml`.

В этой конфигурации раздел `groupTypes` содержит конфигурацию импортируемых групп, а раздел `userTypes` – конфигурацию привязываемых пользователей. Каждый элемент `groupType` должен содержать следующие атрибуты:

- `id` – идентификатор типа группы в рамках конфигурационного файла;
- `xmlField` – имя XML-элементов для групп этого типа;
- `shortName` – шаблон имени группы – в нем можно использовать подстановочные строки `{type}`, `{index}`, `{parent}` (см. ниже);
- `displayName` – шаблон названия группы – в нем также можно использовать подстановочные строки (см. ниже);
- `groupType` – тип группы оргструктуры (`branch` или `role` – подразделение или роль);
- `groupSubType` – подтип группы оргструктуры (тип подразделения – департамент, филиал, … или тип роли – руководитель, рядовой сотрудник);
- `scope` – в каких типах групп может располагаться данная группа (через запятую) или `global` – если в корне оргструктуры.

В атрибутах `shortName` и `displayName` задаются шаблоны для имен группы. При этом `shortName` – это идентификатор группы, а `displayName` – это название группы на естественном языке. В шаблонах можно использовать подстановочные строки:

- `{type}` – заменяется на идентификатор типа группы (задаваемый атрибутом `id`);
- `{index}` – заменяется на уникальный номер группы в рамках своего типа;
- `{parent}` – заменяется на имя родительской группы (в зависимости от контекста либо на `shortName`, либо на `displayName`);
- `{name}` – заменяется на имя группы из входного XML-файла, например, «Бухгалтерия».

Элемент `userType` обычно приводится один и содержит атрибуты, аналогичные атрибутам `groupType`:

- `id` – идентификатор типа пользователя в рамках конфигурационного файла;
- `xmlField` – имя XML-элементов для пользователей этого типа;
- `alfField` – имя свойства, по которому в репозитории ищется пользователь;
- `scope` – в каких типах групп может располагаться данный пользователь (через запятую) – обычно это все группы с `groupType=role`.

При импорте оргструктуры из XML-файла, группы сопоставляются по названию (`displayName`), а пользователи – по полю, указанному в атрибуте `alfField`. Если соответствующая группа не найдена, то она создается. Создание пользователей не осуществляется (обычно создание пользователей осуществляется отдельно – через импорт пользователей из LDAP или из CSV-файла).

Для импорта оргструктуры из XML-файла необходимо загрузить этот файл в Alfresco и вызвать следующий веб-скрипт:

```
http://localhost:8080/alfresco/service/citeck/orgstruct-import?nodeRef=...&from=0&to=100
```

В атрибуте nodeRef указывается идентификатор загруженного XMl-файла, в атрибуте from – номер первой записи для загрузки (включительно, в атрибуте to – номер последней записи для загрузки (не включительно). Собственно загрузка состоит из последовательного вызова веб-скриптов с различными индексами:

```
...&from=0&to=100
...&from=100&to=200
...&from=200&to=300
...&from=300&to=400
...
```

Импорт следует прекратить, когда веб-скрипт возвращает пустую структуру данных, т.е. записей с указанными индексами уже нет.

Данный инструмент импорта является **разовым** (т.е. им нельзя пользоваться для синхронизации оргструктуры на постоянной основе). Первая причина этого в том, что в качестве идентификатора группы используется ее название, которое часто может меняться. В результате для одного и того же подразделения (должности) будет создано несколько разных групп, что приведет к некорректности импортированной оргструктуры. Также в этом подходе не отслеживаются перемещения пользователей и групп: если пользователь (группа) был перемещен в другую группу, то в результате повторного импорта он окажется как в старой группе, так и в новой.

Импорт оргструктуры из Excel

Оргструктуру можно единоразово импортировать из Excel-файла. Для этого он преобразуется в XML-файл и далее производится процедура, описанная в предыдущем разделе. Примерный вид Excel-файла, аналогичный предыдущему XML-файлу следующий:

	A	B	C	D	E	F
1	**company**	**branch**	**department**	**subdivision**	**employee**	**userid**
2	Рога и копыта	Головной офис	Управление	Канцелярия	Делопроизводитель	minintseva
3	Рога и копыта	Головной офис	Управление	Юридический отдел	Делопроизводитель	verdiktov
4	Рога и копыта	Филиал в г. Тула	Бухгалтерия		Главный бухгалтер	zaytseva
5	Рога и копыта	Филиал в г. Тула	Бухгалтерия		Бухгалтер	kuznetsova

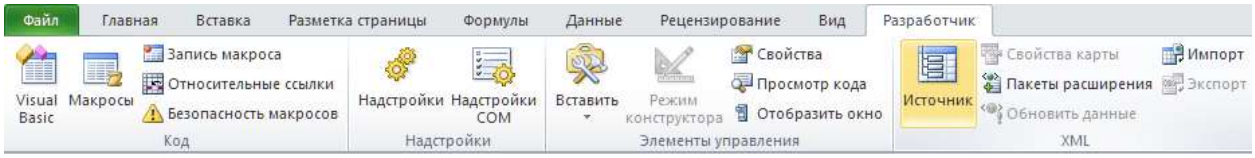
Ниже описано как преобразовать Excel-файл в XML-файл с помощью Microsoft Excel.

Сначала необходимо описать схему XML-файла в формате XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.citeck.ru/orgstruct/test/1.0"
   xmlns:dd="http://www.citeck.ru/orgstruct/test/1.0"
   elementFormDefault="qualified" version="0.1">

    <xs:complexType name="record">
        <xs:sequence>
            <xs:element name="company"     type="xs:string" minOccurs="0" />
            <xs:element name="branch"      type="xs:string" minOccurs="0" />
            <xs:element name="department"  type="xs:string" minOccurs="0" />
            <xs:element name="subdivision" type="xs:string" minOccurs="0" />
            <xs:element name="employee"    type="xs:string" minOccurs="0" />
            <xs:element name="userid"      type="xs:string" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>

    <xs:element name="orgstruct">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="record" type="dd:record"
                            maxOccurs="unbounded" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>

</xs:schema>
```

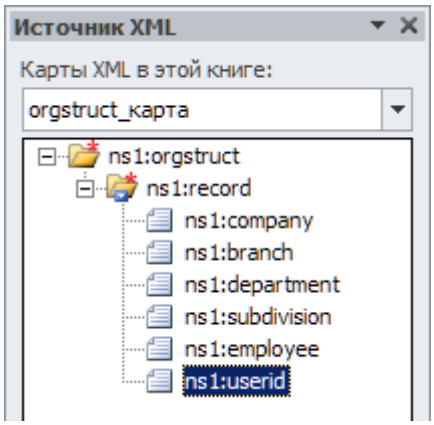
Важно чтобы количество и последовательность колонок в Excel-файле и XSD-схеме совпадало.

После этого ее необходимо подключить в Excel на вкладке ленты «Разработчик» (для включения этой вкладки необходимо перейти в Меню – Файл – Параметры – Настройка ленты и выбрать вкладку «Разработчик»). На вкладке разработчик располагается кнопка «Источник»:



Необходимо перейти в Источник – Карты XML и добавить XSD-схему, сохраненную выше.

После добавления XSD область задач «Источник XML» будет выглядеть так:



Далее нужно нажать на `ns1:record` и выбрать «Сопоставить элемент». После этого таблица станет раскрашенной:

	A	B	C	D	E	F
1	company	branch	department	subdivision	employee	userid
2	Рога и копыта	Головной офис	Управление	Канцелярия	Делопроизводитель	minintseva
3	Рога и копыта	Головной офис	Управление	Юридический отдел	Делопроизводитель	verdiktov
4	Рога и копыта	Филиал в г. Тула	Бухгалтерия		Главный бухгалтер	zaytseva
5	Рога и копыта	Филиал в г. Тула	Бухгалтерия		Бухгалтер	kuznetsova

Далее необходимо сохранить файл в формате XML (XML-данные). Он будет сохранен в следующем виде:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns1:orgstruct xmlns:ns1="http://www.citeck.ru/orgstruct/test/1.0">
    <ns1:record>
        <ns1:company>Рога и копыта</ns1:company>
        <ns1:branch>Головной офис</ns1:branch>
        <ns1:department>Управление</ns1:department>
        <ns1:subdivision>Канцелярия</ns1:subdivision>
        <ns1:employee>Делопроизводитель</ns1:employee>
        <ns1:userid>minintseva</ns1:userid>
    </ns1:record>
    ...
```

Далее нужно убрать в нем подстроки `ns1:` и XML-файл для импорта оргструктуры будет готов: его можно будет использовать для импорта оргструктуры, как описано в предыдущем разделе.