Объектная модель подключаемых модулей

Последние изменения: 2024-03-26

Рассмотрим объектную модель подключаемых модулей. Базовым интерфейсом для всех подключаемых модулей является IConnectivityBase:

[C#]

```
/// <summary>
/// Базовый интерфейс для подключаемых модулей
/// </summary>
public interface IConnectivityBase
{
/// <summary>
/// Уникальный идентификатор модуля.
/// </summary>
string Id
{
get;
}
/// <summary>
/// Флаг, указывающий разрешена или нет работа модуля.
/// </summary>
bool Enabled
{
get;
set;
}
/// <summary>
/// Инициализация (запуск) модуля. После вызова этой функции он должен перейти в  рабочее
состояние
/// </summary>
void Initialize();
/// <summary>
/// Инициализировано соединение или нет.
/// </summary>
bool Initialized
{
get;
}
/// <summary>
/// Функция проверки дополнительных лицензионных ограничений модуля
/// </summary>
param name="supportedSystems">Список внешних модулей, упомянутых в лицензионном
файле</param>
 void
CheckLicenseLimitations(System.Collections.Generic.List<Cleverence.Licensing.LicenseExternalSystem

supportedSystems);
}
```

От интерфейса IConnectivityBase наследуется интерфейс соединения с внешней системой IConnector и интерфейс расширения (плагина) IPlugin.

[C#]

```
/// <summary>
/// Интерфейс соединения с внешней системой. Важное отличие - наличие метода Invoke, позволяющего вызывать любые
какие-то методы из внешней системы
/// </summary>
public interface IConnector: IConnectivityBase
{
/// <summary>
/// Тайм-аут при вызове функций внешней системы через коннектор.
/// </summary>
int Timeout
{
get;
set;
}
/// <summary>
/// Поведение при наступлении тайм-аута. ThrowException - вызывать исключение, ReInvoke - завершить попытку вызова,
переинициализировать подключение и сделать снова вызов.
/// </summary>
TimeoutBehavaior TimeoutBehavaior
{
get;
set;
}
/// <summary>
/// Сообщает, что коннектор сам внутри обрабатывает таймауты, и внешняя обработка не требуется
/// </summary>
bool IsSelfTimeoutBehavior { get; }
/// <summary>
/// Для ActiveMQ коннектора добавляет в аргументы DeviceInfo.
/// </summary>
bool IsSupportDeviceInfoInArgs
{
get;
}
/// <summary>
/// Вызов метода внешней системы.
/// </summary>
/// <param name="methodName">Имя метода.</param>
/// <param name="args">Параметры.</param>
/// <returns>Результат (null, если метод ничего не возвращает).</returns>
object InvokeMethod(string methodName, object[] args);
}
/// <summary>
/// Базовый интерфейс плагина.
/// Немного расширяет стандартный IConnectivityBase явным методом остановки модуля.
/// </summary>
public interface IPlugin: IConnectivityBase
```

```
{
/// <summary>
/// Деинициализация (остановка) модуля. После вызова этой функции он должен остановить
свою работу.
/// </summary>
void Deinitialize();
}
```

Основное отличие между внешним соединением (коннектором) и расширением (плагином) в том, что коннектор имеет функцию InvokeMethod, с помощью которой выполняется вызов внешней системы при запросах с ТСД и при обработке событий сервера. Свойства Timeout, TimeoutBehavaior, IsSelfTimeoutBehavior позволяют установить тайм-аут при вызовах и задать поведение при наступлении тайм-аута. Плагин, в отличии от коннектора, не поддерживает вызовы. И коннектор и плагин запускаются при вызове функции Initialize сервером Mobile SMARTS. Вызов Initialize происходит при старте сервера базы данных Mobile SMARTS (если в настройках базы включена опция «Инициализировать подключения к внешним системам при старте») или при обращении к коннектору (вызов с ТСД или обработка события сервера). Конкретная реализация Initialize определяется особенностями системы, к которой выполняется подключение. Например, может быть выполнено создание COM-объекта для работы со внешней системой и выполнено подключение к базе данных системы с заданным логином/паролем. В случае плагина Initialize запускает модуль в работу. Например, может начаться слежение за папкой или запущен таймер для выполнения периодических операций. Плагин имеет функцию Deinitialize, при вызове которой работа плагина прекращается. Работа плагина может быть остановлена через панель управления (контекстное меню узла плагина -> «Остановить»), также остановка происходит при остановке сервера базы данных Mobile SMARTS.

Еще одним интерфейсом, производным от **IConnectivityBase**, является **IEventsProcessor**:

```
[C#]

/// <summary>
/// Базовый интерфейс для модуля, умеющего обрабатывать события.
/// Для реализации обработки событий необходимо пользоваться
/// атрибутами EventProcessorAttribute, DocumentEventProcessorAttribute
/// </summary>
public interface IEventsProcessor: IConnectivityBase
{
}
```

Интерфейс не добавляет каких-либо функций в **IConnectivityBase**, а служит маркером для модулей, которые используются при обработке событий сервера.

Существует базовая реализация интерфейса **IConnectivityBase**, от которой можно наследоваться при реализации своих коннекторов — класс **ConnectorBase**.

Если разрабатываемый коннектор будет использоваться для обработки событий сервера, рекомендуется в качестве базового использовать другой класс — **ConnectorTypical**, производный от **ConnectorBase**. Данный класс позволяет указывать имена обработчиков событий сервера в свойствах самого коннектора.

[C#]

Код класса приведен не полностью

```
/// <summary>
/// Базовый класс коннектора, позволяющий указывать имена обработчиков событий в свойствах самого коннектора.
/// При обработке событий вызываются функции класса, помеченные атрибутом EventProcessor или DocumentEventProcessor.
/// При разработке новых коннекторов рекомендуется наследоваться от данного класса.
/// </summary>
public abstract class ConnectorTypical : ConnectorBase
{
/// <summary>
/// Имя функции-обработчика события "Документ добавляется".
/// Событие вызывается перед добавлением документа в базу Mobile SMARTS из внешней системы.
/// </summary>
[EventHandlerProperty(EventType.DocumentAdding)]
public string DocumentAddingHandler { get; set; }
.....
/// <summary>
/// Обработчик вызывается перед добавлением документа на сервер из внешней системы.
/// </summary>
/// <param name="di"></param>
/// <param name="documentTypeName"></param>
/// <param name="doc"></param>
/// <returns>Возвращается переданный документ или null, если требуется отклонить добавление документа (будет вызвано исключение и документ не будет сохранен на сервере).
</returns>
[DocumentEventProcessor(DocumentType = "*", EventType = EventType.DocumentAdding)]
public virtual object DocumentAdding(DeviceInfo di, string documentTypeName, Document doc)
...
}
```

Имена обработчиков в панели управления указываются в свойствах коннектора:

Указание обработчиков событий сервера в свойствах коннектора позволяет организовать вызов нескольких обработчиков для одного события. Порядок вызова обработчиков редактируется в панели управления в свойствах узла «События сервера»:

Класс ConnectorTypical содержит виртуальные функции, вызываемые при обработке событий сервера, данные функции помечены атрибутом DocumentEventProcessor для событий документов и атрибутом EventProcessor для остальных событий (номенклатуры и таблиц). Например:

```
[C#]

<summary>
/// Обработчик вызывается сервером при запросе документа с ТСД.
/// </summary>
/// <returns>Возвращается документ (Document), может в сериализованном в xml-строку виде.
Если документ не был получен, то возвращается null. Также может возвращаться InvokeResult.
</returns>
[DocumentEventProcessor(DocumentType = "*", EventType = EventType.GetDocument)]
public virtual object GetDocument(DeviceInfo di, string documentTypeName, string  identity,
GetDocumentMode mode)
```

Атрибут DocumentEventProcessor имеет параметры: DocumentType, позволяет ограничить вызов данного обработчика определенными типами документов, «*» — вызов будет выполнен для всех типов документов, EventType — тип события. Атрибут EventProcessor имеет один параметр: EventType — тип события. При реализации своих обработчиков в производных классах также следует использовать атрибуты DocumentEventProcessor и EventProcessor.

Типы событий, задаваемые перечислением EventType:

События документов
DocumentAdding
Документ добавляется. **Событие, возникающее в процессе добавления документа на сервере. Вызывается при выгрузке документа в базу Mobile SMARTS из внешней системы.**
DocumentAdded
Документ добавлен. **Событие, возникающее после добавления документа на сервере. Вызывается при выгрузке документа в базу Mobile SMARTS из внешней системы.**
DocumentAppointing
Документ назначается пользователю. **Событие о том, что документ готов передаваться на мобильное устройство. Вызывается при запросе с ТСД получения документа для работы.**
DocumentAppointed
Документ назначен пользователю. **Событие о том, что документ захвачен на обработку. Вызывается в момент, когда документ был передан на ТСД для работы с ним.**
DocumentChanged
Документ изменен. **Событие об изменении документа. Вызывается при сохранении документа на сервер в процессе работы на ТСД при использовании в конфигурации Mobile SMARTS действия «Сохранение документа на сервер».**
DocumentFinished
Документ завершен. **Событие о завершении обработки документа. Вызывается при получении сервером завершенного документа с ТСД.**

DocumentReleased
Документ возвращен с ТСД без обработки. Вызывается когда документ с терминала был возвращен пользователем вызовом release (вернуть документ без изменений).
DocumentRemoved
Документ удален. Событие об удалении документа с сервера. Удаление выполняется внешней системой.
ListDocuments
Получить список документов. Вызывается при входе в список документов на ТСД. Используется в паре с событием ПолучитьДокумент для того, чтобы реализовать выбор документов прямо из учетной системы, без предварительной выгрузки.
GetDocument
Получить документ. Вызывается при выборе в списке документов на ТСД того документа, который еще не был выгружен (попал в этот список с помощью события ПолучитьСписокДокументов).
События номенклатуры
ProductNotFound
Получить товар. Событие о том, что продукт был запрошен терминалом по коду (штрихкод, артикул, код) с сервера и не был найден в серверном справочнике.
ProductsNotFound
Получить товары по списку Id. Событие о том, что несколько товаров были запрошены терминалом по идентификаторам с сервера и не были найден в серверном справочнике. Позволяет реализовывать получение сразу нескольких товаров из базы учетной системы без предварительной выгрузки.
GetProductByPartOfName
Получить товар по части наименования. Вызывается при вводе в строку поиска в списке номенклатуры на ТСД. Используется для того, чтобы реализовать поиск номенклатуры по части наименования прямо из учетной системы.

GetCellById
Получить ячейку по идентификатору
GetCellByBarcode
Получить ячейку по штрихкоду
ListProducts
Получить список товаров. Вызывается при входе в список номенклатуры на ТСД. Используется для того, чтобы реализовать выбор номенклатуры прямо из учетной системы, без предварительной выгрузки.
События таблиц
TableRequest
Обработать запрос. Событие вызывается при запросе данных из таблицы, определенной в конфигурации Mobile SMARTS. Позволяет получать строки таблицы он-лайн по запросу с ТСД.

Не нашли что искали?

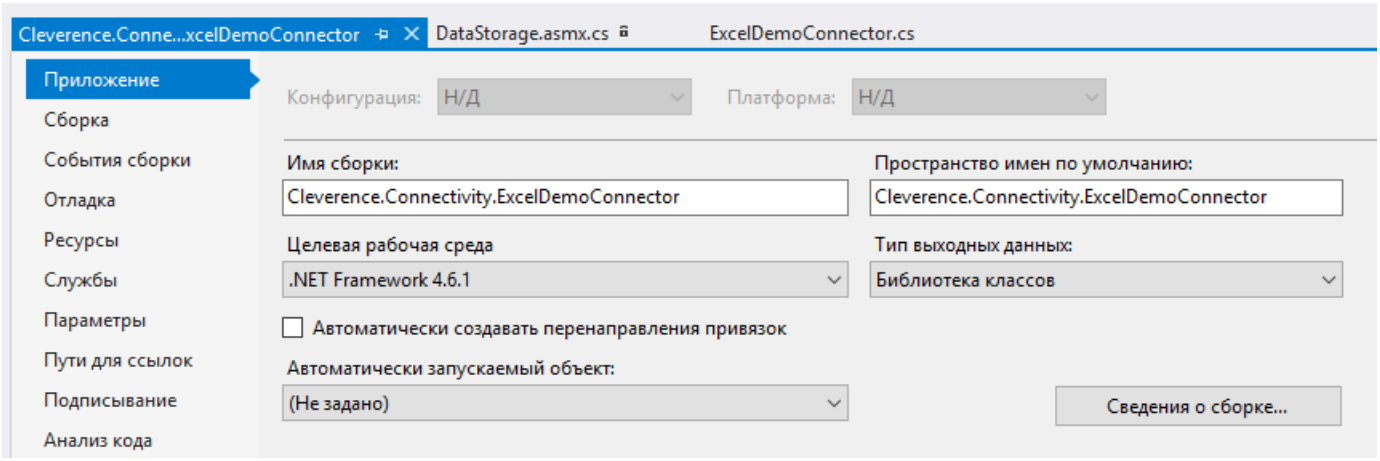
Задать вопрос в техническую поддержку

Разработка собственного коннектора к внешней системе

Последние изменения: 2024-03-26

Для работы коннектора под управлением сервера Mobile SMARTS и настройки параметров коннектора через панель управления создаются два файла dll: первая dll, предназначенная для сервера, размещается в <Папка базы Mobile SMARTS>\ Server\DataService\ bin\. Вторая dll, для панели управления — в <Папка базы Mobile SMARTS>\ Control panel\ Addins. Имена файлов должны иметь вид Cleverence.Connectivity.*.dll (например, Cleverence.Connectivity.MyConnector.dll — для сервера и Cleverence.Connectivity.MyConnector.Panel.dll — для панели управления).

В «Решении» (Solution) в Visual Studio нужно создать два проекта с типом выходных данных «Библиотека классов»:



Скачать заготовку коннектора (Решение Visual Studio 2019 с двумя проектами).

Пространство имен класса коннектора должно начинаться на Cleverence.Connectivity, целевая рабочая среда .NET Framework 4.6.1.

Для серверной версии коннектора в «Ссылки» (Reference Assemblies) нужно добавить следующие dll:

Cleverence.Barcoding.dll

Cleverence.Common.dll

Cleverence.Connectivity.dll

Cleverence.DataCollection.dll

Cleverence.MobileSMARTS.dll

Данные библиотеки находятся по пути <папка установки Mobile SMARTS>\ Server\DataService\ Bin (по умолчанию, C:\ Program Files (x86)\ Cleverence Soft\ Mobile SMARTS\ Server\ DataService\ Bin).

В проект коннектора для панели управления нужно добавить в «Ссылки» (Reference Assemblies):

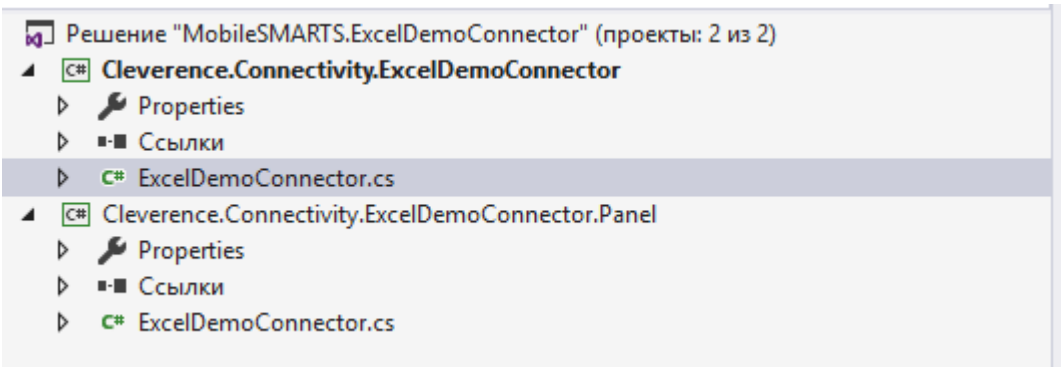
Cleverence.DataCollection.dll

Cleverence.MobileSMARTS.ComConnector.dll

Библиотека Cleverence.MobileSMARTS.ComConnector.dll находится по пути <папка установки Mobile SMARTS>\ Control Panel (по умолчанию C:\ Program Files (x86)\ Cleverence Soft\ Mobile SMARTS\ Control Panel).

Рекомендуется скопировать все нужные dll в отдельную папку рядом с файлом «Решения» (Solution) и добавить ссылки на библиотеки из этой папки.

В каждый из проектов (для сервера и для панели управления) нужно добавить по файлу *.cs для исходного кода класса коннектора.



Класс коннектора должен реализовывать интерфейс IConnector (находится в пространстве имен Cleverence.Connectivity). Можно наследовать класс своего коннектора от класса ConnectorTypical (также из Cleverence.Connectivity), который реализует IConnector и, кроме того, содержит свойства для указания имен обработчиков событий сервера и виртуальные функции для вызова обработчиков.

Обе версии класса коннектора (для сервера и для панели управления) должны иметь одинаковое имя и находится в одинаковых пространствах имен (namespace), начинающихся на Cleverence.Connectivity (например, Cleverence.Connectivity.MyConnector).

При наследовании от ConnectorTypical в серверной версии коннектора следует перегрузить функции: Initialize (выполняет инициализацию коннектора), Deinitialize (выполняет деинициализацию), InvokeMethod (выполняет вызов во внешнюю систему). Именно InvokeMethod реализует основной функционал коннектора по работе с внешней системой. Также следует перегрузить свойство Initialized (возвращает признак, инициализирован ли коннектор, true — инициализирован, false — нет). При необходимости, могут быть перегружены функции обработки событий сервера (например, GetProduct — получение товара при запросе с ТСД, DocumentFinished — на сервер с ТСД попал завершенный документ и др.). Если не перегружать функции обработки событий, при возникновении событий будет вызываться функция InvokeMethod, в которую передается имя обработчика события, указанное в настройках и соответствующие событию аргументы (см. События сервера).

Версия коннектора для панели управления также наследуется от ConnectorTypical. Перегружать какие-либо функции в этом случае не нужно. Для подключения коннектора к базе внешней системы обычно требуются определенные настройки (адрес базы внешней системы, имя пользователя/ пароль и т.п). Для того, чтобы иметь возможность редактировать нужные настройки и чтобы настройки сохранялись, требуется добавить свойства в классы коннектора для панели управления и сервера. Заготовка коннектора:

[C#]

```
Серверная часть:
namespace Cleverence.Connectivity.DemoConnector
{
public class DemoConnector : ConnectorTypical
{
public DemoConnector()
{
}
public string MyProperty
{
get;
set;
```

```
set;
}
public override bool Initialized
{
get
{
throw new NotImplementedException();
}
}
public override void Initialize()
{
throw new NotImplementedException();
}
public override void Deinitialize()
{
throw new NotImplementedException();
}
public override object InvokeMethod(string methodName, object[] args)
{
throw new NotImplementedException();
}
}
}
```

Часть для панели управления:

```
namespace Cleverence.Connectivity.DemoConnector
 {
public class DemoConnector : ConnectorTypical
{
public DemoConnector()
{
}
public string MyProperty
{
get;
set;
}
}
}
```

В серверной части нужно реализовать Initialized, Initialize, Deinitialize, InvokeMethod. Для примера добавлено свойство MyProperty.

Собранную dll серверной версии размещаем в <Папка базы Mobile SMARTS>\Server\DataService\bin\, версию для панели управления в <Папка базы Mobile SMARTS>\Control panel\Addins.

Сервер Mobile SMARTS в целях безопасности выполняет проверку цифровой подписи загружаемых сборок. Если после размещения неподписанного файла dll в <Папка базы Mobile SMARTS>\Server\DataService\bin\> перезапустить службу сервера Mobile SMARTS, в логе сервера базы данных dataserver_*.log (в C:\ProgramData\Cleverence\Logs) появится сообщение:

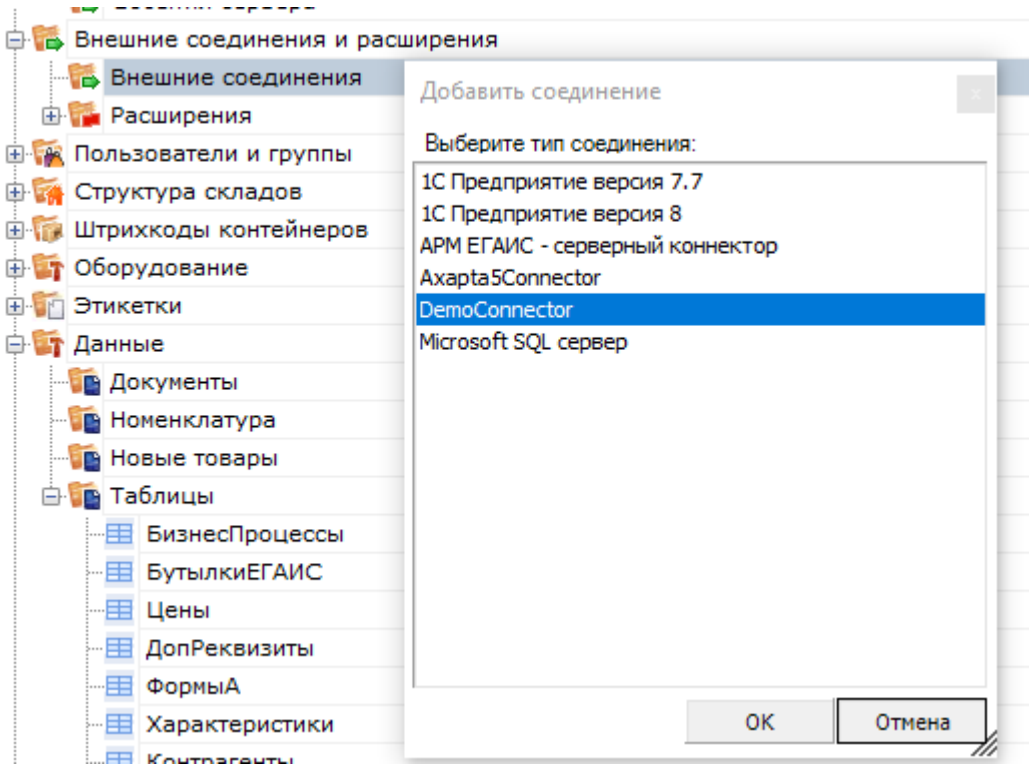
2019-08-05 10:08:08.6537|ERROR|NLogger.WriteNlogEvent| Коннекторы не загружены! Обнаружена неподписанная сборка: C:\ProgramData\Cleverence\Базы Mobile SMARTS\e3945857-308f-4829-92e2-720dc11d1bec\Server\DataService\bin\Cleverence.Connectivity.DemoConnector.dll

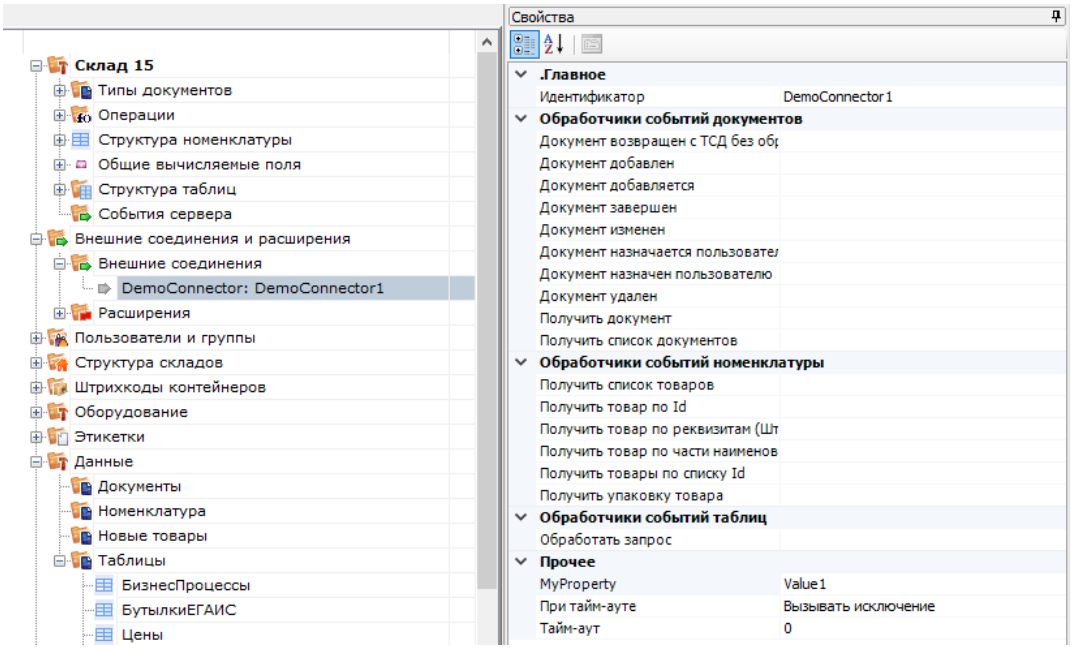
Видим, что dll коннектора не загрузилась. В некоторых случаях загрузка неподписанной сборки приводит

к остановке сервера. Для того, чтобы можно было проверить работу коннектора и выполнить отладку, сервер нужно запустить с ключом Debug из командной строки (службу сервера перед этим нужно остановить):

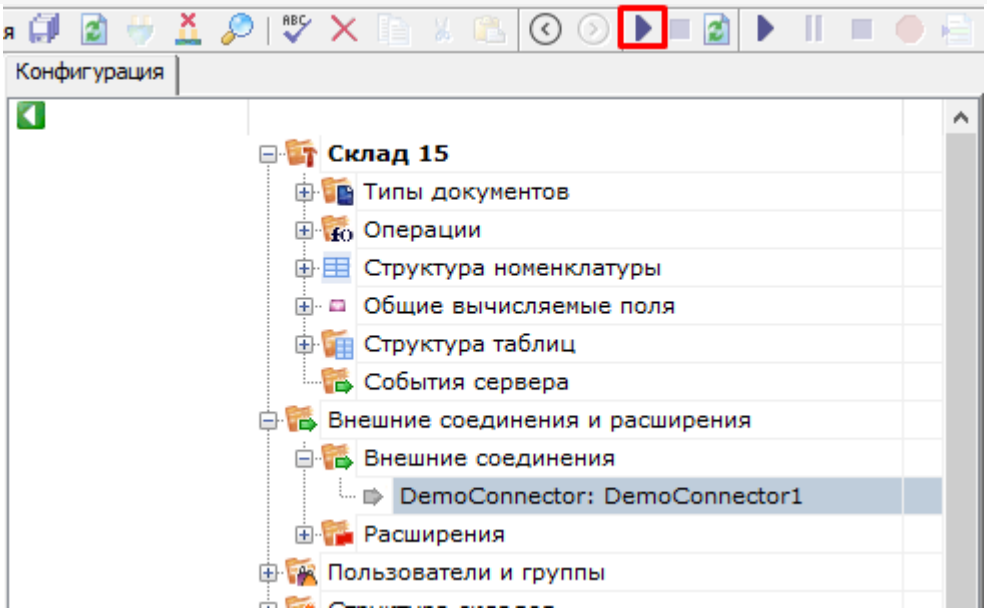
C:\Program Files (x86)\Cleverence Soft\Mobile SMARTS\Server\Cleverence.MobileSMARTS.Server.exe /debug

После этого в панели управления можно добавить коннектор в конфигурацию:





Настройка параметров коннектора выполняется через панель свойств. Когда настройка выполнена, сохраним конфигурацию. После этого можно запустить коннектор с помощью кнопки «Пуск»:



Если требуется отладка кода коннектора, рядом с файлом коннектора в <Папка базы Mobile SMARTS>\Server\DataService\bin\ следует разместить файл *.pdb (см. Тестирование и выпуск разработанного коннектора). Когда отладка закончена, обратитесь в техническую поддержку «Клеверенс» для подписания dll.

Скачать заготовку коннектора (Решение Visual Studio 2019 с двумя проектами).

Не нашли что искали?

Задать вопрос в техническую поддержку

Пример разработки коннектора к внешней системе

Последние изменения: 2024-03-26

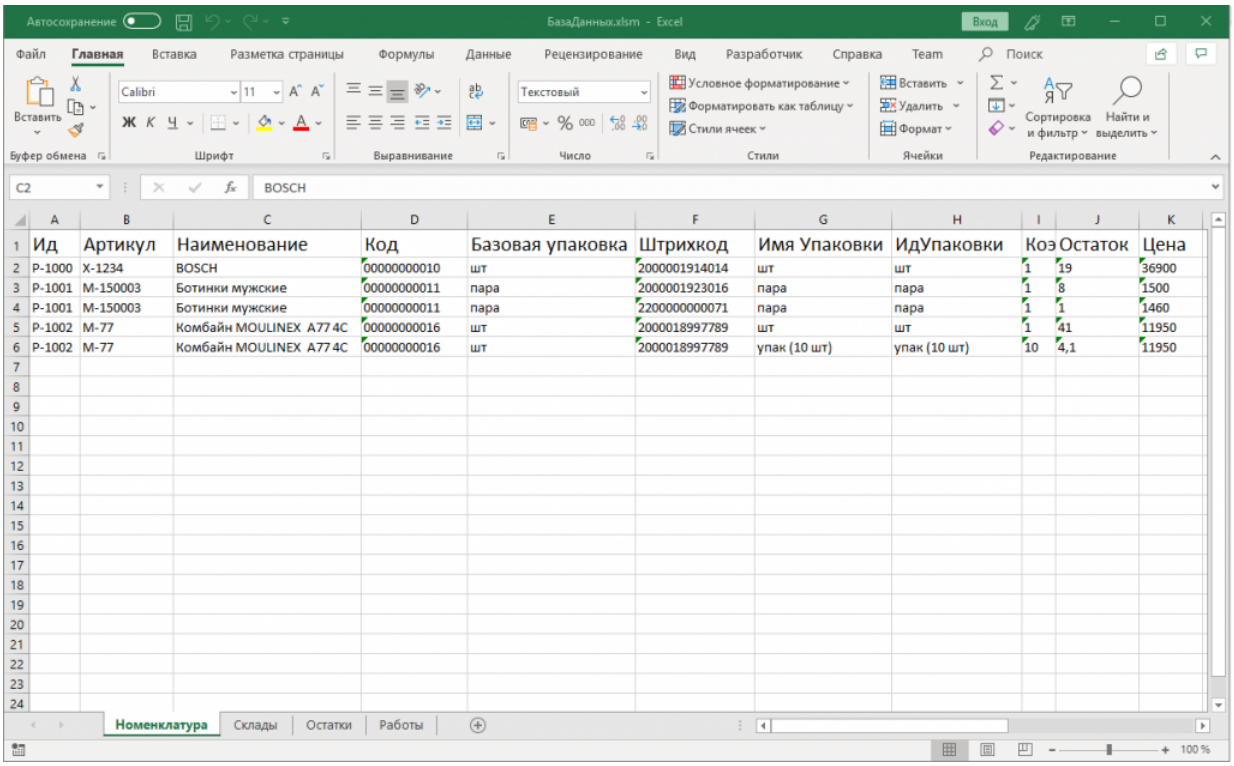
Для примера разработаем коннектор, который будет выполнять получение данных из файла Excel онлайн по запросам с ТСД. В примере будет рассмотрена, в том числе, перегрузка функций-обработчиков событий сервера для получения номенклатуры, документов и обработки запросов к таблицам.

Постановка задачи

Требуется разработать коннектор, который будет по запросам с ТСД получать данные из файлов Excel. Используются следующие справочники: «Номенклатура», «Склады», «Остатки». Должна быть обеспечена работа с документами «Поступление», на ТСД должен отображаться список документов для работы, подготовленных на сервере. Пользователь ТСД выбирает документ, принимает товар, после завершения работы данные записываются в исходный документ. Используется база Mobile SMARTS «Магазин 15».

Подготовка данных

Справочники будут храниться в одном файле (книге) Excel на листах: «Номенклатура», «Склады», «Остатки». Файл со справочниками имеет фиксированное название «БазаДанных.xlsm» (книга Excel с поддержкой макросов) и располагается в заданной папке (путь к папке указывается в настройках коннектора). Файлы документов «Поступления» будут находиться в этой же папке и называться «Поступление №00001.xlsx», «Поступление №00002.xlsx» и т. д. В архиве с исходниками данные находятся в папке Xlsx, состав полей см. в файлах.



	A	B	C	D	E	F	G	H	I	J	K
1	Ид	Артикул	Наименование	Код	Базовая упаковка	Штрихкод	Имя Упаковки	ИдУпаковки	Коэ	Остаток	Цена
2	P-1000	X-1234	BOSCH	00000000010	шт	2000001914014	шт	шт	1	19	36900
3	P-1001	M-150003	Ботинки мужские	00000000011	пара	2000001923016	пара	пара	1	8	1500
4	P-1001	M-150003	Ботинки мужские	00000000011	пара	2200000000071	пара	пара	1	1	1460
5	P-1002	M-77	Комбайн MOULINEX A77 4C	00000000016	шт	2000018997789	шт	шт	1	41	11950
6	P-1002	M-77	Комбайн MOULINEX A77 4C	00000000016	шт	2000018997789	упак (10 шт)	упак (10 шт)	10	4,1	11950

Разработка и тестирование коннектора

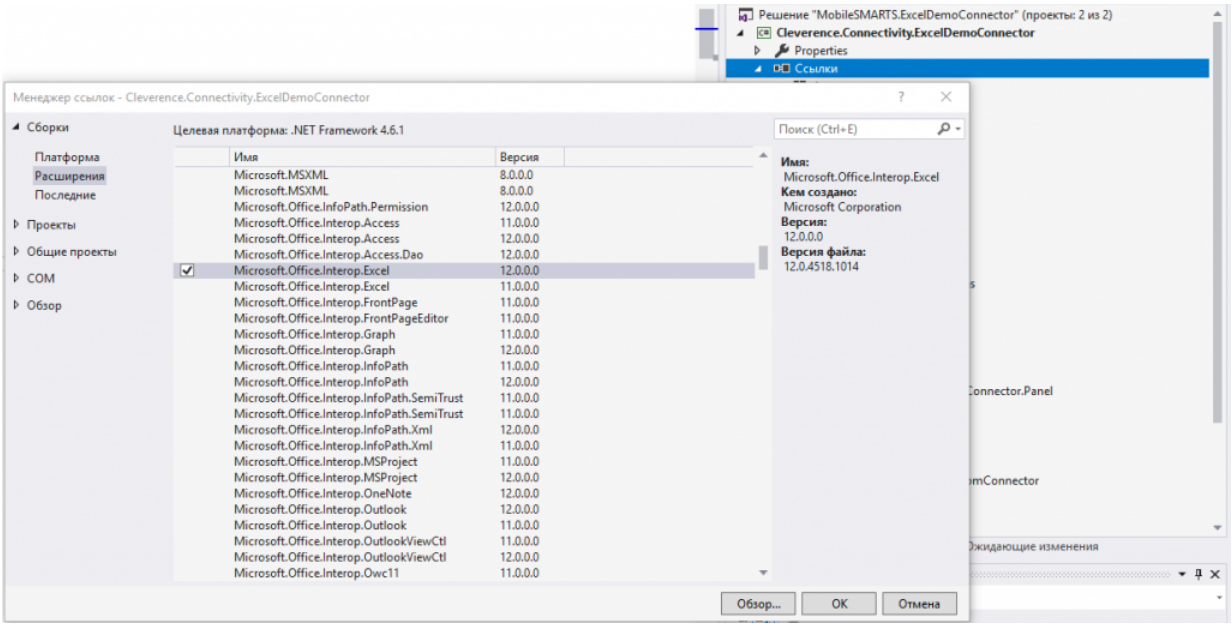
В Visual Studio создаем «Решение» (solution) и два проекта в нем:

Cleverence.Connectivity.ExcelDemoConnector (для сервера) и
Cleverence.Connectivity.ExcelDemoConnector.Panel (для панели управления).

Добавляем в ссылки (Reference assemblies) в серверный проект Cleverence.Connectivity.ExcelDemoConnector: Cleverence.Common.dll, Cleverence.Connectivity.dll, Cleverence.DataCollection.dll, Cleverence.MobileSMARTS.dll.

В проект для панели управления Cleverence.Connectivity.ExcelDemoConnector.Panel: Cleverence.DataCollection.dll, Cleverence.MobileSMARTS.ComConnector.dll.

Для работы с файлом Excel будем использовать COM-объект Excel.Application (на ПК должен быть установлен Microsoft Excel), добавим в ссылки в проект для сервера Microsoft.Office.Interop.Excel:



Добавим в каждый из проектов по файлу ExcelDemoConnector.cs, в котором будет находится класс коннектора ExcelDemoConnector. Базовым классом для ExcelDemoConnector является ConnectorTypical.

Добавим свойство, с помощью которого можно задать путь к папке с файлами Excel, используемыми для обмена данными:

[C#]

```
public string DatabaseFolder
{
get;
set;
}
```

Свойство должно быть как в серверном варианте коннектора, так и в варианте для Панели управления. Других настроек для нашего коннектора не требуется, займемся разработкой серверной части, которая будет выполнять обмен данными с Excel.

Добавим приватное поле Excel.Application excel для хранения ссылки на COM-объект Excel. Теперь первым делом нам нужно реализовать, функцию Initialize, которая переводит коннектор в рабочее состояние:

```
[C#]

public override void Initialize()
{
this.Dispose();
if (!this.Enabled)
throw new InvalidOperationException(this.GetType().Name + " is not enabled.");
this.excel = new Excel.Application();
this.excel.Visible = false;
}
```

Вызываем Dispose, чтобы выполнить деинициилиазацию, если ранее коннектор был инициализирован. Если вызовы коннектора запрещены через данель управления (Enabled == false), вызываем исключение. Если вызовы разрешены, создаем COM-объект.

Деинициализация:

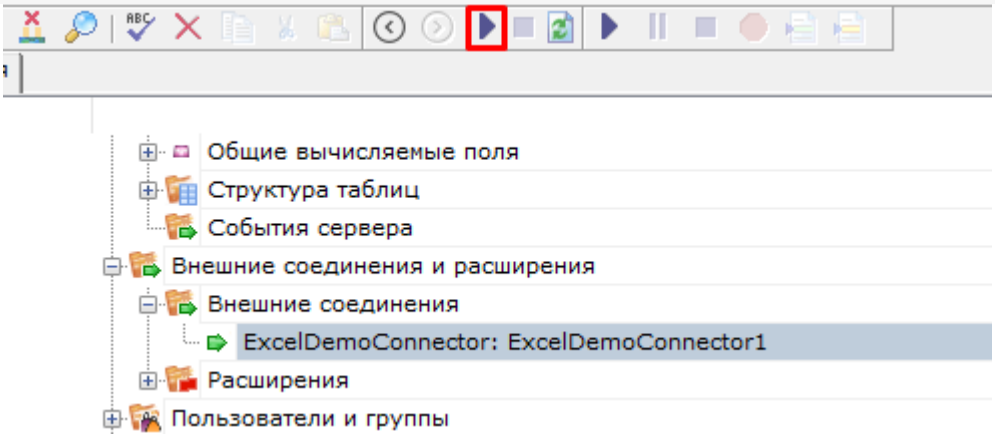
```
[C#]

public override void Deinitialize()
{
if (this.excel != null)
{
this.excel.Quit();
this.excel = null;
}
}
Признак того, что коннектор инициализирован:
public override bool Initialized
{
get
{
return this.excel != null;
}
}
```

Этого достаточно, чтобы проверить запуск коннектора через панель управления.

Соберем решение в Visual Studio. Развернем базу «Магазин 15 Расширенный» или «Мегамаркет». Скопируем dll для сервера в <Папка базы Mobile SMARTS>\Server\DataService\bin\, для панели управления в <Папка базы Mobile SMARTS>\Control panel\Addins. Запустим сервер в режиме отладки из командной строки: C:\Program Files (x86)\Cleverence Soft\Mobile SMARTS\Server\Cleverence.MobileSMARTS.Server.exe/debug.

Добавим коннектор в конфигурацию, сохраним и проверим, что выполняется запуск.



Реализуем получение номенклатуры онлайн из файла Excel. Когда товар запрашивается терминалом на сервере по каким-либо реквизитам (Ид, Штрихкод, Артикул) и товар не найден в выгруженном справочнике номенклатуры, выполняется вызов обработчика события «Получить товар»

В случае наследования от ConnectorTypical обработчик события может быть реализован как во внешней системе (вызов через InvokeMethod функции из внешней системы по имени), так непосредственно в коннекторе. Для реализации обработчика в самом коннекторе нужно перегрузить функцию GetProduct:

```
[C#]

[EventProcessor(EventType = EventType.ProductNotFound)]
public override object GetProduct(Cleverence.Warehouse.DeviceInfo di, string productId, string
packingId,
SearchProductMode searchMode, Cleverence.Barcoding.BarcodeData barcodeData)
{
CheckInit();
return GetProductInternal(productId, packingId, searchMode);
}
```

Не забываем указать атрибут EventProcessor (EventType = EventType.ProductNotFound). Реализацию поиска товаров по заданным реквизитам в зависимости от режима (searchMode) см. в GetProductInternal.

Событие «Получить список товаров» возникает, когда пользователь на ТСД заходит в список номенклатуры (выбор товара по 0 в действии «Выбор номенклатуры») и в базе нет выгруженного справочника. Перегрузим в коннекторе функцию GetProductsList:

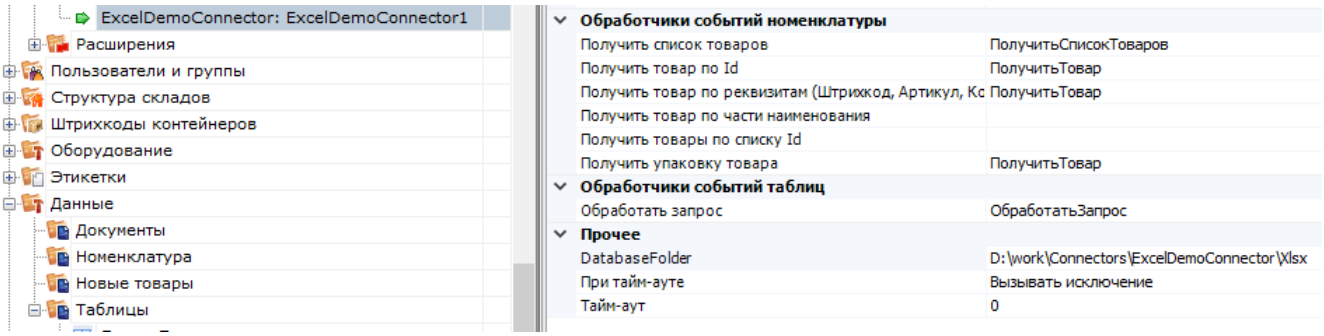
```
[C#]

[EventProcessor(EventType = EventType.ListProducts)]
public override object GetProductsList(DeviceInfo di, string searchStr)
{
CheckInit();
return GetProductsListInternal(searchStr);
}
```

Функция возвращает PackedProductCollection со всем списком номенклатуры.

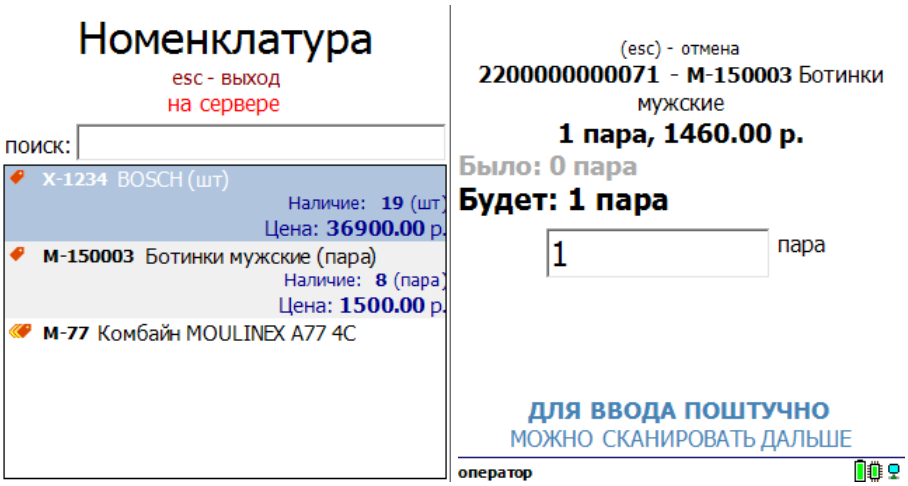
Проверим, как происходит получение номенклатуры. В настройках коннектора нужно указать путь к папке, в которой находится файл Excel со справочниками (DatabaseFolder). Нужно заполнить обработчики событий

номенклатуры: «Получить список товаров», «Получить товар по Id», «Получить товар по реквизитам» (штрихкод, артикул, код), «Получить упаковку товара».



Задание имен обработчиков требуется для того, чтобы указать, что коннектор имеет подписку на определенные события. При вызове обработчика через InvokeMethod указанные имена обработчиков передаются в InvokeMethod.

Проверить получение номенклатуры можно с помощью клиента Mobile SMARTS для ПК. Для получения списка товаров заходим в «Просмотр справочников -> Товары». Проверить получение по штрихкоду можно, например, в операции «Сбор штрихкодов»:



Реализуем получение списка документов. Обработчик события «Получить список документов» вызывается, когда пользователь на ТСД заходит в список выбора документов по кнопке типа документа.

```
[C#]

[DocumentEventProcessor(EventType = EventType.ListDocuments)]
public override object GetDocumentsList(DeviceInfo di, string documentTypeName)
{
...
}
```

Функция возвращает DocumentDescriptionCollection со списком описаний документов (DocumentDescription), готовых для отдачи на ТСД. Список получаем на основе имен файлов xlsx в папке с данными.

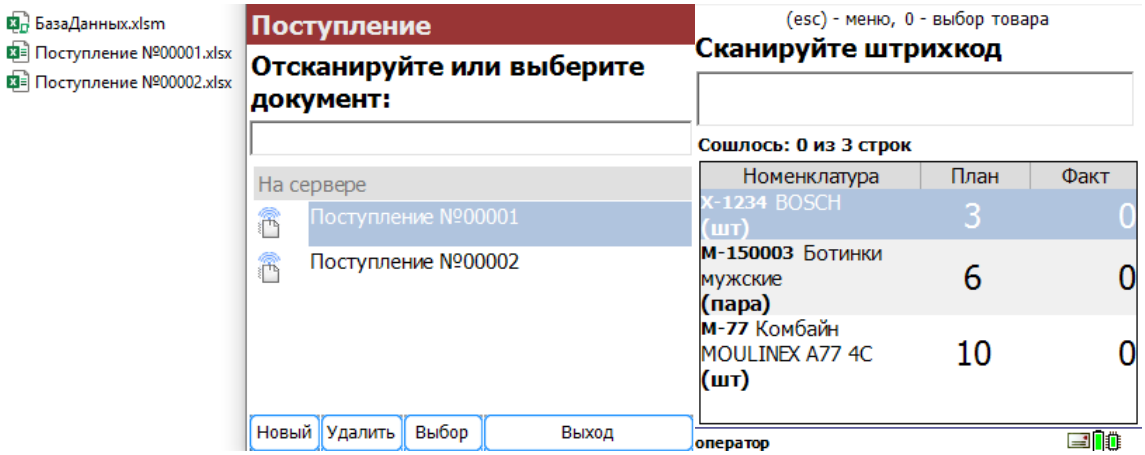
Для получения документа в работу на ТСД нужно реализовать обработчик события «Получить документ». Обработчик вызывается, когда пользователь на ТСД выбирает документ из списка, полученного с помощью события «Получить список документов», или сканирует штрихкод документа в окне выбора документов.

```
[C#]

[DocumentEventProcessor(EventType = EventType.GetDocument)]
public override object GetDocument(DeviceInfo di, string documentTypeName, string identity,
GetDocumentMode mode)
{
...
}
```

Функция возвращает объект Document, если документ найден по переданным параметрам.

Проверим работу на примере документов «Поступление»:



Для загрузки завершенного на ТСД документа используем событие «Документ завершен».

```
[C#]

[DocumentEventProcessor(EventType = EventType.DocumentFinished)]
public override object DocumentFinished(DeviceInfo di, string documentTypeName, Document doc)
{
CheckInit();
string fileName = GetFileNameByDocId(doc.Id, documentTypeName);
if (!string.IsNullOrEmpty(fileName))
{
if(LoadDocumentFromTerminal(fileName, doc))
{
MessageCenter.AddNewMessage(string.Format("Документ '{0}' загружен.", doc.Name), null,
doc.UserId, false);
BaseRuntimeContext.Current.DocumentsManager.Delete(doc);
return doc;
}
}
return base.DocumentFinished(di, documentTypeName, doc);
}
```

В данной функции после загрузки документа на ТСД отправляется сообщение с помощью MessageCenter.AddNewMessage. Завершенный документ удаляется из базы Mobile SMARTS с помощью BaseRuntimeContext.Current.DocumentsManager.Delete (doc).

Реализуем получение данных онлайн при запросах к таблицам Mobile SMARTS. Например, в конфигурации «Магазин 15» есть таблицы «Склады», «Остатки» и др. На ТСД может выполняться получение списка всех складов, запрос остатков определенного товар и др. Если таблица хранится на сервере, в настройках таблицы включен поиск во внешней системе и задан обработчик события «Обработать запрос», то сервер вызывает указанный обработчик. Перегрузим в нашем коннекторе функцию ProcessTableRequest.

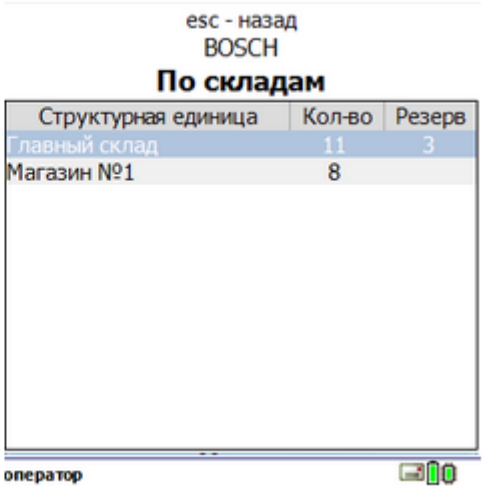
```
[C#]

[EventProcessor(EventType = EventType.TableRequest)]
public override object ProcessTableRequest(DeviceInfo di, DocumentQuery dq, DocumentTableInfo
tableInfo)
{
...
}
```

В функцию передается: DocumentQuery dq — объект запроса, в свойстве WhereRootElement содержится корень синтаксического дерева запроса. Обойдя дерево, можно сформировать запрос в том виде, который ожидает внешняя система. DocumentTableInfo tableInfo — описание таблицы, в свойстве Name содержится имя таблицы из конфигурации Mobile SMARTS. В простейшем случае можно возвращать все строки запрошенной таблицы, не накладывая условие из DocumentQuery, сервер Mobile SMARTS сам выполнит выборку данных по заданным условиям. Однако, в случае реальной работы с большими объемами данных, так делать не рекомендуется.

Функция возвращает коллекцию строк RowCollection.

Проверим получение складов и остатков:



Кроме обработки событий сервера, есть возможность вызова произвольных функций внешней системы, которые будут возвращать некоторые данные на ТСД или выполнять какую-то работу во внешней системе. В конфигурации Mobile SMARTS для этого используется действие «Вызов внешней системы». В коннекторе должна быть реализована функция InvokeMethod.

В нашем случае InvokeMethod будет вызывать макрос Excel с указанным именем, передавая полученные с ТСД аргументы:

[C#]

```
public override object InvokeMethod(string methodName, object[] args)
{
CheckInit();
string dbPath = Path.Combine(DatabaseFolder, DbFileName);
Excel.Workbook workbook = this.excel.Workbooks.Open(dbPath);
try
{
return RunMacro(methodName, args);
}
finally
{
workbook.Close();
}
}
```

Добавим в книгу Excel лист «Работы», на котором будет таблица с количеством собранных сотрудниками заказов за неделю:

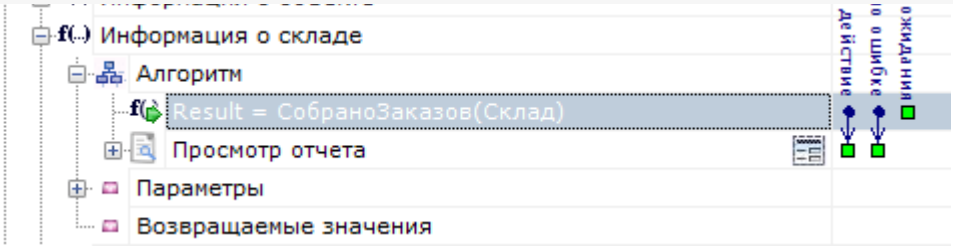
	A	B	C
1	Комплектовщик	Склад	Собрано заказов
2	Фахуртдинов	Главный склад	352
3	Петров	Главный склад	278
4	Коршунов	Магазин №1	94
5	Мухамедов	Магазин №2	65

На VB в Excel напишем функцию, которая будет возвращать таблицу с колонками «Комплектовщик», «Собрано заказов» по переданному складу.

[C#]

```
Function СобраноЗаказов(Склад As String)
Dim sh_src As Worksheet
Dim storageConnector As Object, rowCollection As Object, row As Object
Dim picker As String
Dim ordCnt As Integer
Dim rng As Excel.Range
Set storageConnector = CreateObject("Cleverence.Warehouse.StorageConnector")
Set rowCollection = CreateObject("Cleverence.Warehouse.RowCollection")
Set sh_src = Worksheets("Работы")
.....
СобраноЗаказов = storageConnector.ToXml(rowCollection)
End Function
```

Функция возвращает объект Cleverence.Warehouse.RowCollection, сериализованный в xml. В конфигурации Mobile SMARTS сделаем вызов данной функции при просмотре информации о складе:



В действии «Просмотр отчета» выведем полученную таблицу:

Не нашли что искали?

Задать вопрос в техническую поддержку

Тестирование и выпуск разработанного коннектора к внешней системе

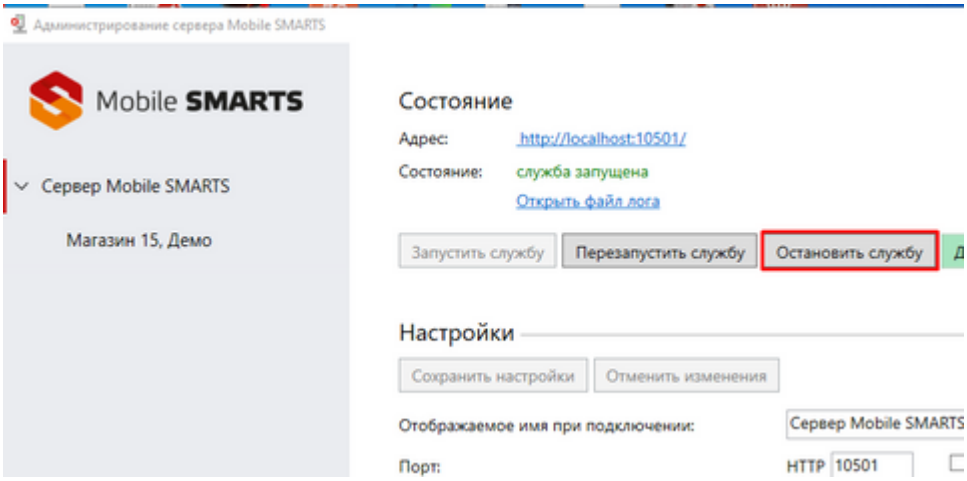
Последние изменения: 2024-03-26

В связи с повышением требований к безопасности сервера все разработанные коннекторы должны пройти процедуру проверки и подписания у «Клеверенс».

Попытка использования неподписанного коннектора в «продуктивном» режиме приводят к полной остановке сервера с записью в лог файле «Коннекторы не загружены! Обнаружена неподписанная сборка: xxxx.dll», либо «Коннекторы не загружены! Ошибка проверки подписи dll: xxxx.dll».

Для отладки коннектора при разработке следует использовать запуск сервера в тестовом режиме.

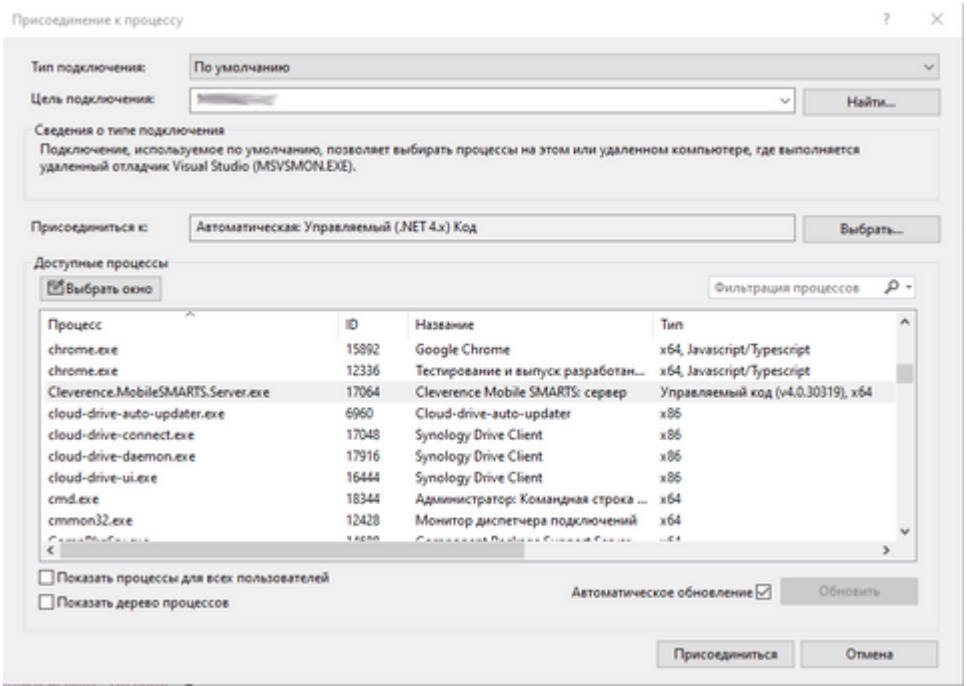
Для этого остановите сервис сервера:



И запускайте его из командной строки с параметром /debug:

c:\Program Files (x86)\Cleverence Soft\Mobile SMARTS\Server\Cleverence.MobileSMARTS.Server.exe /debug

Далее Вы можете просто подключаться отладчиком Visual Studio к запущенному процессу и вести отладку вашего коннектора.



После завершения разработки вашего коннектора создайте запрос в техническую поддержку «Клеверенс» о его проверке и подписании.

Не нашли что искали?

Задать вопрос в техническую поддержку