Быстрая интеграция с 1С: Предприятие

* [PHP](https://habr.com/ru/hub/php/),
* [Python](https://habr.com/ru/hub/python/),
* [.NET](https://habr.com/ru/hub/net/)
* [Из песочницы](https://habr.com/ru/sandbox/)

В данной статье я расскажу о наиболее простом, на мой взгляд, способе интеграции сторонних приложений с конфигурациями 1С. Статья будет интересна в первую очередь разработчикам, пишущим на **.Net Core**, **PHP** и **Python**.  
  
Способов интеграции с 1С известно много, этому даже посвящена прекрасная [статья](https://habr.com/ru/company/1c/blog/308420/) от самой [компании 1С](https://habr.com/ru/company/1c/). Из нее вы в частности узнаете, что 1С поддерживает механизмы web-сервисов, а значит мы можем реализовать свой собственный сервис на стороне 1С, и, как следствие, свою собственную ORM-библиотеку на стороне клиентского приложения. Об одной из таких и библиотек и пойдет речь далее.

Описание методики в общих чертах

На стороне 1С

Все начинается с того, что на конфигурацию 1С ставится расширение «Бром», добавляющее новый web-сервис. Расширение находится в свободном доступе и распространяется по лицензии (MIT). Само расширение не привязано к конкретной модели данных, и поэтому может быть установлено на любые конфигурации c поддержкой режима совместимости 8.3.10 или выше.  
  
Как только расширение установлено, необходимо настроить права пользователей, которые будут иметь доступ к методам web-сервиса, а также опубликовать конфигурацию на web-сервере, чтобы добавленный сервис был доступен по http(s) протоколу. На стороне 1С более ничего не требуется.

На стороне клиентского приложения

На стороне клиентского приложения подключается пакет Brom, например для .Net Core это можно сделать командой:

Install-Package Brom -Version 1.0.1-beta08

Или для Python:

pip install brome

После установки пакета достаточно создать объект клиента, через которого будет осуществляться взаимодействие с удаленной конфигурацией 1С. Здесь и далее я буду приводить код на C#, но на PHP и Python он аналогичен. И так, создать клиента можно одной командой:

dynamic клиент = new БромКлиент(@"

Публикация = http://mydomain.com/publication\_name;

Пользователь = 1c\_user\_name;

Пароль = 1c\_user\_pass

");

В конструкторе достаточно указать адрес опубликованной конфигурации 1С, и данные пользователя, которому мы выдали права на доступ к расширению. Как только клиент создан мы можем приступать к самому интересному.

Вызов процедур и функций 1С

Через созданного бром-клиента мы можем вызывать процедуры и функции определенные в 1С. При этом вызываемые методы должны быть серверными и содержаться либо в глобальном контексте, либо в серверных модулях (общих модулях или модулях менеджеров). Например, так выглядит вызов функции глобального контекста «ЧислоПрописью»:

string числоПрописью = клиент.ЧислоПрописью(2547, "Л = fr\_FR");

Второй параметр — это форматная строка с указанием локализации (французский язык). Параметры передаются в естественном виде и не требуют какого-либо дополнительного преобразования или упаковки.  
  
А вот так будет выглядеть вызов функции «НайтиПоКоду» модуля менеджера справочника:

var доллар = клиент.Справочники.Валюты.НайтиПоКоду(840);

Тут мы вызвали функцию через модуль менеджера справочника «Валюты». Результатом вызова будет объект типа «СправочникСсылка». Теперь полученную ссылку на объект можно передать в качестве параметра в другую функцию:

var курсыВалют = клиент.РаботаСКурсамиВалют.ПолучитьКурсВалюты(доллар, DateTime.Today);

На этот раз мы обратились к общему модулю «РаботаСКурсамиВалют» и вызволи его метод «ПолучитьКурсВалюты».  
  
Библиотека поддерживает работу со сложными типами данных, поэтому возможно вызывать методы, которые принимают на вход или возвращают: Ссылки, Массивы, Структуры, ТаблицыЗначений, ДеревьяЗначений, системные перечисления и пр… Некоторые классы специально реализованы в клиентской библиотеки для упрощения работы с 1С.

Работа со ссылками

Ссылки на объекты позволяют не только передавать указатель на объект 1С, но и получать данные самого объекта. Работа со ссылками на клиентской стороне так же проста как и в 1С. Например:

var заказ = клиент.Документы.ЗаказКлиента.НайтиПоНомеру("ТД00-000018", new Date(2017, 1, 1));

var датаЗаказа = заказ.Дата;

var контрагент = заказ.Контрагент;

var иннКонтрагента = заказ.Контрагент.ИНН;

foreach (var стр in заказ.Товары) {

Console.WriteLine((стр.Номенклатура, стр.Количество));

}

Здесь мы нашли ссылку на документ через модуль менеджера и получили значения полей документа и его табличной части «Товары». После первого обращения к полю объекта все его данные загружаются с сервера 1С и хранятся на клиенте до удаления ссылки сборщиком мусора.  
  
Ссылку на объект можно получить и на клиентской стороне без обращения к серверу. Для этого достаточно знать уникальный идентификатор объекта:

var заказ = клиент.Документы.ЗаказКлиента.ПолучитьСсылку(new Guid("5a32b6ab-4661-11e9-912a-38d547755ef7"));

Так же просто можно получить ссылки на предопределенные элементы коллекций:

var ставкаНДС = клиент.Перечисления.СтавкиНДС.НДС18\_118;

Редактирование объектов

Имея ссылку на объект, мы можем редактировать данные объекта. Для этого достаточно создать контекст объекта:

var заказОбъект = заказ.ПолучитьОбъект();

заказОбъект.Дата = DateTime.Today;

заказОбъект.Номер = "ТД00-000055";

заказОбъект.Товары.Очистить();

var стр = заказОбъект.Товары.Добавить()

стр.Номенклатура = клиент.Справочники.Номенклатура.НайтиПоКоду("000000104");

стр.Количество = 3;

заказОбъект.Записать(РежимЗаписиДокумента.Проведение);

В данном примере мы создали контекст документа через ссылку на документ, заполнили некоторые поля, добавили строку в табличную часть «Товары» и записали документ в режиме проведения.  
  
Если необходимо создать новый объект, то это тоже возможно:

*// Создаем контекст группы справочника*

var группаОбъект = клиент.Справочники.Номенклатура.СоздатьГруппу();

группаОбъект.Наименование = "Новая группа";

группаОбъект.Записать();

*// Создаем контекст элемента справочника*

var товарОбъект = клиент.Справочники.Номенклатура.СоздатьЭлемент();

товарОбъект.Родитель = группаОбъект.Ссылка;

товарОбъект.Наименование = "Новый товар";

товарОбъект.Артикул = "T-00012321";

*// Записываем объект справочника*

товарОбъект.Записать();

*// Получаем ссылку на созданный объект*

var товарСсылка = товарОбъект.Ссылка;

Здесь мы создали новую группу в справочнике «Номенклатура», затем создали элемент справочника и поместили его в созданную группу.

Формирование выборок

Как и любая приличная ORM, бром-клиент позволяет формировать выборки из различных объектных коллекций 1С. Выборка — это коллекция ссылок на объекты коллекции, которые удовлетворяют набору условий выборки. Для формирования выборки достаточно создать объект «Селектор»:

var группаМебель = клиент.Справочники.Номенклатура.НайтиПоНаименованию("Мебель");

var селектор = клиент.Справочники.Номенклатура.СоздатьСелектор();

селектор.

Выбрать("Наименование, Код, Производитель, Производитель.ИНН").

Где("ЭтоГруппа", false).

Где("Ссылка", группаМебель, ВидСравнения.ВИерархии).

Упорядочить("Производитель").

Упорядочить("Наименование", НаправлениеСортирвки.Убывание);

foreach (var ссылка in селектор) {

Console.WriteLine("Наименование: {0}; Код: {1}, Производитель: {2}; ИНН: {3}",

ссылка.Наименование,

ссылка.Код,

ссылка.Производитель,

ссылка.Производитель.ИНН

);

}

*// Сохраняем результат выборки в массив для последующего использования*

var результат = текСелектор.ВыгрузитьРезультат();

В данном примере мы получили выборку, в которой содержатся элементы справочника «Номенклатура», находящиеся иерархически в группе «Мебель». Мы указали, что кроме самих ссылок необходимо загрузить данные некоторых полей. Благодаря этому, данные указанных полей будут загружены одним запросом, и обращение к ним не будет приводить к дополнительным серверным вызовам.

Выполнение запросов

Чаще всего данных, хранящихся в одной коллекции, становится недостаточно, и нам требуется получить данные, сформированные сложным запросом. Для выполнения запросов в клиентской библиотеке предусмотрен специальный класс «Запрос». Работа с запросами на стороне клиентского приложение очень похожа работу на стороне 1С:

var запрос = клиент.СоздатьЗапрос(@"

ВЫБРАТЬ

Номенклатура.Ссылка КАК Ссылка,

Номенклатура.Код КАК Код,

Номенклатура.Наименование КАК Наименование

ИЗ

Справочник.Номенклатура КАК Номенклатура

ГДЕ

Номенклатура.Артикул = &Артикул

");

запрос.УстановитьПараметр("Артикул", "Т-0001");

var результат = запрос.Выполнить();

foreach (var стр in результат) {

Console.WriteLine((стр.Код, стр.Наименование));

}

Здесь мы создали простейший запрос с параметром, который выбирает данные из справочника «Номенклатура». В качестве параметра мы передали строковый артикул элемента. В общем случае значением параметра могут быть также ссылки, системные перечисления и даже массивы. В качестве результата выполнения запроса нам вернулась «ТаблицаЗначений», этот класс реализован на клиентской стороне. В данном примере мы вывели поля строк таблицы с помощью цикла.  
  
На стороне 1С все запросы выполняются через построитель запросов, благодаря этому можно указывать в качестве текста не только готовый запрос, но и шаблон запроса, содержащий разметку для построителя:

var запрос= клиент.СоздатьЗапрос(@"

ВЫБРАТЬ ПЕРВЫЕ 5

ЦеныНоменклатурыСрезПоследних.Номенклатура КАК Номенклатура,

ЦеныНоменклатурыСрезПоследних.Характеристика КАК Характеристика,

ЦеныНоменклатурыСрезПоследних.Цена КАК Цена

{ВЫБРАТЬ

Номенклатура.\*}

ИЗ

РегистрСведений.ЦеныНоменклатуры.СрезПоследних(

{(&Период)},

{ (Номенклатура).\*, (Характеристика).\*}

) КАК ЦеныНоменклатурыСрезПоследних

{ГДЕ

ЦеныНоменклатурыСрезПоследних.Цена}

{УПОРЯДОЧИТЬ ПО

Номенклатура.\*,

Характеристика.\*}

");

*// Добавляем дополнительное поле запроса с указанием синонима поля*

запрос.ДобавитьПоле("Номенклатура.Производитель.Наименование", "Бренд");

*// Добавляем дополнительный отбор по цене*

запрос.ДобавитьУсловиеОтбора("Цена", 100, ВидСравнения.БольшеИлиРавно);

*// Указываем дополнительное упорядочение по двум полям*

запрос.ДобавитьУпорядочение("Номенклатура.Производитель");

запрос.ДобавитьУпорядочение("Характеристика", НаправлениеСортировки.Убывание);

var результат = запрос.Выполнить(ОбходРезультатаЗапроса.Прямой);

В данном примере мы указали шаблонизированный запрос, настройки которого могут динамически изменятся на клиентской стороне по мере необходимости. Благодаря этому поля, отборы и сортировки могут указываться в теле основного кода программы, а не внутри тела запроса.  
  
Метод «Выполнить» принимает опциональный параметр «тип обхода результатов». Если указать тип обхода «По группировкам» или «По группировкам с иерархией», то вместо таблицы значений, метод вернет дерево значений.  
  
Также вместо метода «Выполнить» можно вызывать метод «ВыполнитьПакет» (для выполнения сразу нескольких запросов). В этом случае будет возвращен массив таблиц или массив деревьев, в зависимости от типа обхода.

Выполнение фрагментов кода

В некоторых экзотических случаях может потребоваться выполнить определенный фрагмент кода непосредственно на стороне 1С. Для этого в бром-клиенте предусмотрен метод «Выполнить». Метод принимает на вход текст, содержащий исполняемый код и один опциональный параметр, который будет доступен в коде в переменной «Параметр»:

var суммаЧиселВМассиве = клиент.Выполнить(@"

Результат = 0;

Для Каждого Значение Из Параметр Цикл

Результат = Результат + Значение;

КонецЦикла;

", new double[] { 45, 67, 12.56, 11.9 });

В данном примере мы выполнили фрагмент кода, который суммирует числа в массиве. Сам массив был передан в качестве параметра и помещен в переменную «Параметр». Результат вычисления был помещен в переменную «Результат». Если в исполняемом коде заполняется переменная с таким именем, то ее значение на момент окончания исполнения возвращается в качестве результата функции «Выполнить».  
  
Возможность выполнения фрагментов кода регулируется отдельной ролью доступа в расширении. Рекомендуется отключать эту роль, а требуемые фрагменты выполнять в виде заранее определенных функций на стороне 1С.

Преимущества недостатки методики

К преимуществам описанной методики безусловно следует отнести:

* **Кроссплатформенность**. Все взаимодействие построено на протоколах SOAP и HTTP, а их реализация есть во всех популярных платформах для разработки;
* **Простота кода**. Код на стороне клиентского приложения практически идентичен коду на стороне 1С;
* **Встроенные механизмы сериализации**. Нам не требуется упаковывать и распаковывать данные, чтобы обмениваться ими с 1С;
* **Поддержка работы со ссылками**. Мы имеем простой доступ к объектам 1С через ссылки;
* **Поддержка специфичных для 1С типов данных**. Мы можем обмениваться с 1С таблицами, деревьями, структурами и прочими сложными конструкциями;
* **Доступ к контексту приложения**. Мы имеем доступ не только к данным БД, но и имеем возможность вызывать методы определенный на стороне 1С, а также имеем доступ к состоянию сеанса.

Недостатки у данной методики тоже имеются:

* **Низкая скорость передачи данных**. Так как в основе SOAP протокола лежит XML-сериализация, то передача больших объемов данных требует времени на обмен избыточным трафиком, а также на упаковку и распаковку данных. Обмен данными через COM-соединение выглядит в данном контексте предпочтительнее;
* **Удобный клиент доступен не во всех платформах**. Если вы не программист .Net Core, PHP или Python, то бром-клиента придется реализовывать самостоятельно, используя механизмы SOAP, что, вообще говоря, трудоемко;
* **Ограничения языка запросов 1С**. Так как вся работа с БД происходит через механизм запросов 1С, присутствуют некоторые ограничения. Например, вы не можете реализовать классический постраничный вывод данных, т.к. в языке запросов 1С отсутствуют механизмы постраничного разбиения.

Сравнение с OData

Из вышеупомянутой [статьи](https://habr.com/ru/company/1c/blog/308420/) фирмы 1С вы можете узнать, что в 1С: Предприятие реализован доступ к данным по стандартизированному протоколу OData. По этой причине было бы глупо не упомянуть и его тоже.   
Вот краткая сравнительная таблица:

|  | **Бром** | **OData** |
| --- | --- | --- |
| Получение данных коллекций со сложными условиями отбора и сортировки | + | + |
| Постраничный вывод данных коллекций |  | + |
| Добавление и редактирование данных | + | + |
| Поддержка работы с типами 1С: Предприятие (ссылки, таблицы, деревья и пр.) | + |  |
| Выполнение произвольных запросов | + |  |
| Вызов серверных процедур и функций | + |  |
| Выполнение произвольного серверного кода | + |  |
| Готовые инструменты визуализации / пользовательские интерфейсы |  | + |
| Передача данных в формате JSON |  | + |

Видно, что данные методики имеют свои преимущества и недостатки и в общем случае не являются взаимозаменяемыми.

Заключение

Надеюсь, данная обзорная статья поможет вам в дальнейшем быстро и просто создавать порталы, личные кабинеты и сервисы, тесно интегрированные с учетными системами на базе 1С: Предприятие. Подробную информацию о бром-компонентах вы можете найти в официальной документации, здесь приведен лишь краткий обзор основных особенностей