# Практическая работа №6. Получение данных из внешних источников

## Введение

Веб-сервис — обладающая уникальным веб-адресом (URL) программная система, построенная на базе открытых протоколов/стандартов и использующаяся для обмена данными между приложениями или системами. Программные приложения, написанные на разных языках программирования и работающие на разных платформах, могут использовать веб-сервисы для обмена данными по компьютерным сетям, таким как Интернет.

Веб-сервис обеспечивает связь между различными приложениями с использованием открытых стандартов, таких как:

* HTML — для обмена Web-страницами;
* XML — для описания данных;
* SOAP — для обмена структурированными сообщениями;
* WSDL — для описания Web-сервисов и доступа к ним;
* JSON — для обмена текстовыми данными.

Основное преимущество веб-сервисов — универсальность. Они работают на базе стандартных технологий, не зависящих от поставщиков приложений, могут использоваться в любых операционных системах, серверах приложений, языках программирования и т.д.

Сегодня веб-сервисы получили широкое распространение. Компании используют их для интеграции информационных систем и доступа клиентов или партнеров к своим ресурсам. Например, банк может создать веб-сервис для предоставления данных о курсах валют и акций, условиям по кредитам и вкладам. Торговая компания — веб-сервис для знакомства клиентов с ассортиментом товаров и ценами.

## REST API

API (Application Programming Interface) — язык, на котором приложения общаются между собой. С помощью API одно приложение может использовать возможности другого приложения. Например, интернет-магазин может вызывать банковские сервисы для оплаты покупок.

Описание правил такого языка называется спецификацией, а порции данных, которыми обмениваются приложения — сообщениями. Сообщения обычно идут парами запрос-ответ. Например, интернет-магазин отправляет запрос банковскому приложению, передавая ему реквизиты свои и покупателя, а также сумму для оплаты. А банковское приложение возвращает ответ, в котором сообщается, прошла ли оплата успешно. Ответ может быть более развернутым — например, чтобы покупатель видел движение своего заказа, интернет-магазин периодически отправляет запросы в службу доставки, в ответах получает данные о текущем статусе и местоположении заказа и передает эти сведения покупателю.

REST API — это не один конкретный протокол взаимодействия, а архитектурный стиль. Он описывает, как разработчику следует спроектировать интерфейс для взаимодействия своего приложения с другими. Если продолжить аналогию с естественным языком, то REST API описывает грамматику.

Для доступа клиентских приложений к данным, хранящимся на сервере, в REST API используются такие методы как POST, GET, PUT и DELETE. При этом обмен сообщениями осуществляется обычно по протоколу HTTP(S).

Главная особенность REST API — обмен сообщениями без сохранения состояния. Каждое сообщение самодостаточное и содержит всю информацию, необходимую для его обработки. Сервер не хранит результаты предыдущих сессий с клиентскими приложениями. Это обеспечивает гибкость и масштабируемость серверной части, позволяет поддерживать асинхронные взаимодействия и реализовывать алгоритмы обработки любой сложности. Кроме того, такой формат взаимодействия является универсальным — он не зависит от технологий, используемых на клиенте и на сервере, и не привязывает разработчиков к определенному провайдеру.

Как это работает? Клиент отправляет запрос на сервер. Сервер аутентифицирует клиента и проверяет его права, затем обрабатывает запрос и возвращает ответ клиенту.

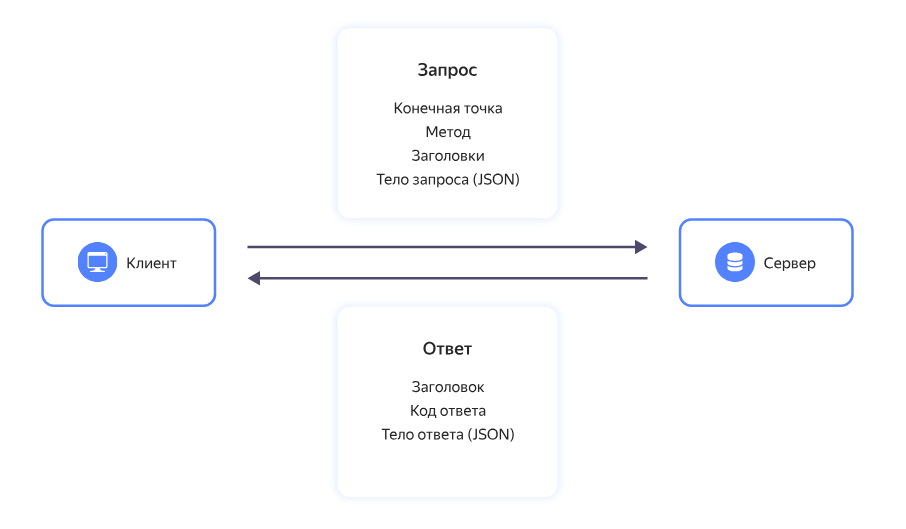


Рисунок 1. Способ обмена данными в REST

Как правило, для взаимодействия между клиентом и сервером достаточно четырех методов:

* GET — получение информации об объекте (ресурсе);
* POST — создание, а иногда и изменение объекта (ресурса);
* PUT — изменение объекта (ресурса). Обычно используется для регулярного обновления данных;
* DELETE — удаление информации об объекте (ресурсе).
* Реже используются методы LIST, UPDATE, PATCH и др.

Запрос REST API от клиента к серверу всегда состоит из следующих элементов:

* **Конечная точка** (endpoint) — адрес, по которому отправляется запрос.

Один и тот же объект (ресурс) может иметь несколько конечных точек.

Например, чтобы отправить запрос на сервер службы доставки Best Delivery, может использоваться конечная точка best-delivery.com/orders. Чтобы посмотреть список заказов, можно использовать конечную точку best-delivery.com/orders/list, а чтобы разместить новый заказ — best-delivery.com/orders/create.

* **Параметры** — делятся на параметры пути и параметры запроса.

Например, в запросе best-delivery.com/orders/{userId}/list {userId} — это параметр пути. Вместо {userId} нужно подставить идентификатор конкретного пользователя, тогда запрос вернет список заказов этого пользователя.

В запросе best-delivery.com/orders/list?orderId=123456 orderId — это параметр запроса. Такой запрос вернет информацию о конкретном заказе.

В одном запросе может быть несколько параметров пути и несколько параметров запроса. Параметры запроса соединяются между собой символом &. Например, запрос best-delivery.com/orders/shop/{shopId}/users/{userId}/list?top=10&sortDate=DESC вернет список заказов в магазине {shopId} для пользователя {userId}, причем список будет содержать только 10 последних заказов, т.к. он отсортирован по убыванию даты заказа.

* Заголовки (headers) — в заголовках определяется формат передаваемых данных, спецификация и версия протокола обмена и другая информация, необходимая для корректной обработки запроса.

Если для выполнения запроса требуется аутентификация, в заголовке передаются сведения о пользователе — логин, токен и т.п. Заголовки не отображаются в пути запроса.

* Тело запроса (body) — данные для обработки, как правило в формате JSON.

Например, запрос для службы доставки может содержать номер заказа, адрес, телефон для связи и интервал доставки, примерно так:

{“orderId”:123456, “address”:”119021, Москва, ул. Льва Толстого, 16”, “phone”:”+74957397000”, “time\_interval”:”9:00-11:00”}.

После выполнения REST API запроса сервер вернет клиентскому приложению ответ. Он включает код ответа, заголовки и тело ответа.

Как и в запросе, заголовки в ответе также определяют формат передаваемых данных, спецификацию и версию протокола обмена, и другие сведения, которые помогут клиентскому приложению правильно прочитать и понять ответ.

Тело ответа — это информация, которую запрашивал клиент. Ответ тоже чаще всего передается в формате JSON. Но тело ответа может быть и пустым.

Код ответа — это признак успешности выполнения запроса. Для унификации используются стандартные коды ответа. Они представляют собой трехзначные числа. Ответы, начинающиеся с цифры 1, обозначаются 1xx, и т.п.

Ответы вида 1хх — информационные.

Ответы вида 2хх говорят об успешном выполнении запроса. Например:

* 200 – ОК. Если клиентом были запрошены какие-либо данные, то они находятся в заголовке или теле сообщения.
* 201 – OK. Создан новый ресурс.

Ответы вида 3xx обозначают перенаправление или необходимость уточнения. Например:

* 300 — на отправленный запрос есть несколько вариантов ответа. Чтобы получить нужный вариант, клиент должен уточнить запрос.
* 301 — запрашиваемый адрес перемещен.
* 307 — запрашиваемый адрес временно перемещен.

Ответы вида 4хх говорят о том, что при выполнении запроса возникла ошибка, и это ошибка на стороне клиента. Например:

* 400 – Bad Request. Запрос некорректный.
* 401 – Unauthorized. Запрос требует аутентификации пользователя.
* 403 – Forbidden. Доступ к сервису запрещен.
* 404 – Not found. Ресурс не найден.

Ответы вида 5хх говорят об ошибке на стороне сервера. Например:

* 503 — сервис недоступен.
* 504 — таймаут (превышено допустимое время обработки запроса).

## CURL для GET и POST запросов на получение данных

cURL — это утилита командной строки для передачи данных с или на удаленный сервер с использованием одного из поддерживаемых протоколов. Он установлен по умолчанию в macOS и большинстве дистрибутивов Linux.

cURL используется разработчиками для тестирования API , просмотра заголовков ответов и выполнения HTTP-запросов.

**Выполнение запроса GET**

Чтобы выполнить запрос GET с помощью Curl, выполните команду curl, за которой следует целевой URL-адрес. Curl автоматически выбирает метод запроса HTTP GET, если не используется параметр командной строки -X, --request или -d. Целевой URL-адрес передается в качестве первого параметра командной строки. Чтобы добавить дополнительные HTTP-заголовки, используйте параметр командной строки -H. Curl автоматически добавляет заголовок запроса Accept: \*/\*, если не передаются дополнительные заголовки, который сообщает серверу, что клиент Curl может принимать данные в любом формате.

*curl* [*https://reqbin.com/echo*](https://reqbin.com/echo)

Чтобы сделать GET-запрос с HTTP-заголовками, используйте параметр командной строки -H. Вы можете передать столько заголовков HTTP с запросом Curl GET, сколько захотите, используя параметр командной строки -H несколько раз.

*curl https://reqbin.com/echo*

*-H "Cache-Control: must-revalidate"*

*-H "Pragma: no-cache"*

*-H "Expires: 0"*

Для получения данных в формате JSON с помощью Curl необходимо передать серверу HTTP-заголовок "Accept: application/json". Если вы не передадите этот заголовок, сервер может автоматически выбрать наиболее подходящий тип данных для вашего клиента и вернуть данные в другом формате. Ниже приведен пример получения JSON из URL-адреса ReqBin:

*curl https://reqbin.com/echo/get/json -H "Accept: application/json"*

**Выполнение запроса POST**

Общая форма команды curl для выполнения запроса POST следующая:

*curl -X POST [options] [URL]*

Параметр -X указывает, какой метод HTTP-запроса будет использоваться при связи с удаленным сервером. Тип тела запроса указывается его заголовком Content-Type.

Чтобы создать запрос POST, используйте параметр -F , за которым следует пара field=value . В следующем примере показано, как отправить POST-запрос в форму с полями «имя» и «электронная почта»:

*curl -X POST -F 'name=linuxize' -F 'email=linuxize@example.com' https://example.com/contact.php*

Другой способ сделать запрос POST — использовать параметр -d . Это заставляет curl отправлять данные с использованием Content-Type application/x-www-form-urlencoded Content-Type.

*curl -X POST -d 'name=linuxize' -d 'email=linuxize@example.com' https://example.com/contact.php*

Если параметр -d используется более одного раза, вы можете объединить данные с помощью символа & :

*curl -X POST -d 'name=linuxize&email=linuxize@example.com' https://example.com/contact.php*

Чтобы установить определенный заголовок или Content-Type, используйте параметр -H . Следующая команда устанавливает тип запроса POST на application/json и отправляет объект JSON:

*curl -X POST -H "Content-Type: application/json"*

*-d '{"name": "linuxize", "email": "linuxize@example.com"}'*

*https://example/contact*

Чтобы отправить файл с помощью curl , просто добавьте символ @ перед местоположением файла. Файл может быть архивом, изображением, документом и т. д.

*curl -X POST -F 'image=@/home/user/Pictures/wallpaper.jpg' http://example.com/upload*

## REST запросы и структуризация данных в Loginom

В демонстрационном примере <https://examples.loginom.ru/json/> показано как можно отправлять REST-запросы на сервер напрямую из Loginom, получать ответ от сервера и структурировать полученную информацию в формат таблицы. Для настройки подключения в сценарии необходимо использовать вкладку «Подключения» определенного модуля сценария Loginom.

Перед тем как использовать в сценариях узлы взаимодействия с SOAP-сервисами/REST-сервисами и импорта/экспорта из бизнес-приложений, хранилищ данных, баз данных или иных источников, в пакете необходимо создать подключение к источнику данных. Подключение содержит в себе все необходимые параметры для соединения с источником данных, такие как пароль/логин, расположение источника и др. Созданное подключение в дальнейшем может использоваться в пакете многократно.

Для каждого модуля в составе пакета можно настроить свой список подключений. Для создания/редактирования подключений необходимо отобразить в панели навигации структуру пакета и выбрать пункт *Подключения* соответствующего модуля (см. рисунок 2).

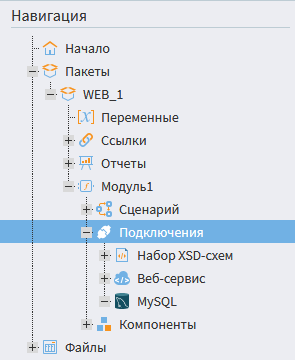


Рисунок 2. Выбор пункта "Подключения" в панели навигации

В результате в левой части экрана в виде дерева отобразится перечень возможных для создания подключений (см. рисунок 2 область 1) — дерево классов подключений.

Подключение можно создать двумя способами:

* перетащить нужное подключение из дерева (1) в область подключений (2);
* дважды кликнуть по выбранному подключению в дереве.

В области подключений появится новое подключение (3), и откроется диалог редактирования параметров подключения.

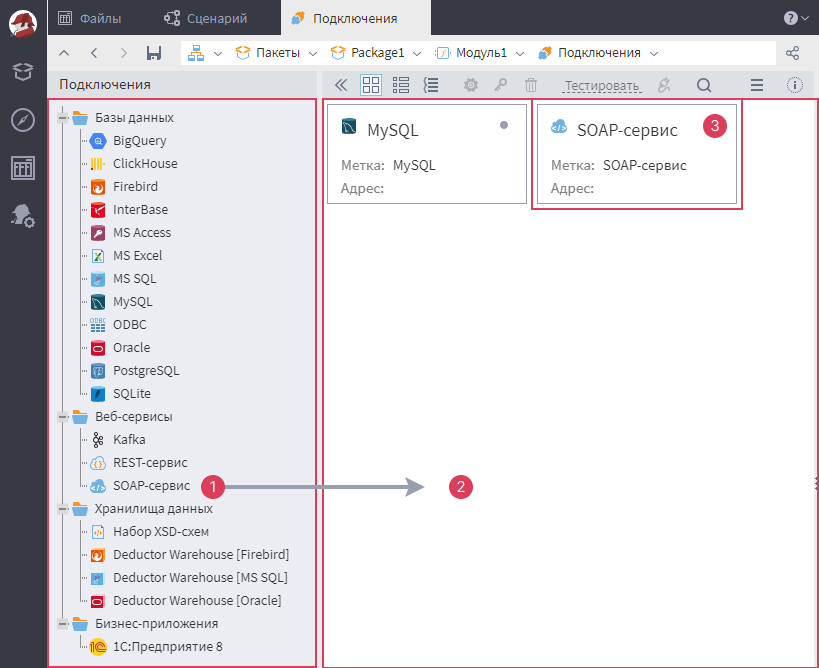


Рисунок 3. Создание нового подключения

После того, как требуемые параметры будут заданы, подключение может быть протестировано.

Затем подключение должно быть перенесено в соответствующий сценарий в том же модуле.

Созданное подключение необходимо поместить в сценарий, для этого необходимо создать узел, выходными данными которого будут являться параметры подключения к источнику данных.

Для создания узла:

Открыть сценарий для редактирования и развернуть панель Подключения, в которой содержится перечень всех созданных в пакете подключений.

Выделить необходимое подключение в панели и, вызвав кликом правой кнопки мыши его контекстное меню, воспользоваться одним из предложенных способов (см. рисунок 1):

Добавить ссылку на Подключение в Сценарий — в сценарий будет добавлен Узел-ссылка (1) на подключение. Также создать узел-ссылку можно при помощи функции Drag-and-drop, перетащив мышью выбранное подключение в область построения сценария.

Добавить узел Подключения в Сценарий — в сценарий будет добавлен производный узел (2), унаследованный от выбранного подключения. Производный узел будет иметь те же настройки, что и выбранное подключение, однако, их возможно переопределить в мастере настройки узла (при этом настройки подключения, от которого был унаследован узел, не изменятся).

Перейти к подключению — переход к окну с доступными подключениями.

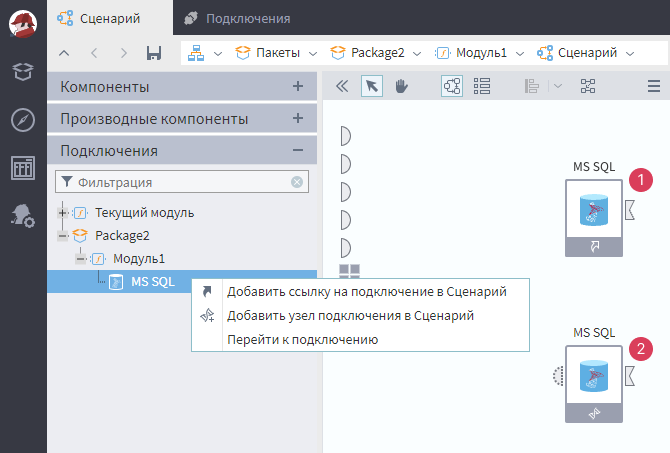


Рисунок 4. Создание узла подключения в сценарии

Параметры подключения к источнику данных используются узлами импорта/экспорта данных.

## Самостоятельное задание

В данной самостоятельной работе студенту необходимо выполнить несколько заданий как на получение данных в терминале Linux в гостевой ОС с помощью встроенной утилиты CURL, так и на получение и структуризацию данных в Loginom с использованием инструментов отправки запросов и преобразования данных.

1. С помощью утилиты CURL в гостевой ОС получить описание вакансии с ID 92400841 на **основе GET запроса** к сервису HH API (*https://api.hh.ru/vacancies/<vacancy\_id>*, документация к просмотру вакансии: <https://api.hh.ru/openapi/redoc#tag/Vakansii/operation/get-vacancy>). Сохраните поток вывода в отдельный файл с названием, содержащим ID вакансии. **(1 балл).**
2. С помощью утилиты CURL в гостевой ОС получить описание вакансии с другим любым ID, в котором найдется вакансия на **основе GET запроса** к сервису HH API (*https://api.hh.ru/vacancies/<vacancy\_id>*, документация к просмотру вакансии: <https://api.hh.ru/openapi/redoc#tag/Vakansii/operation/get-vacancy>). Сохраните поток вывода в отдельный файл с названием, содержащим ID вакансии. **(1 балл).**
3. С помощью утилиты CURL в гостевой ОС получить набор данных Obesity\_random\_sample с сервиса *https://loginom-1.mirea.ru/lgi/rest/obesity\_data/Obesity\_random\_sample* на основе **метода POST**. Посмотреть примеры запросов и ответов с сайта можно по ссылке: https://loginom-1.mirea.ru/lgi/rest/obesity\_data/help.

Передайте свое значение параметра random\_number в файле данных. Для метода создайте отдельный файл json, данные которого необходимо отправить на сервер. Параметр CURL -d “@data.json” позволит отправить данные из файла и получить по ним ответ. Ответ сервиса из потока вывода в терминал перенаправить в текстовый файл с названием **obesity.json** **(1 балл).**

1. С помощью Loginom получите данные свечей для акций компаний MSFT или другой любой интересующей вас компании, представленной на Yahoo Finance.   
   Для этого пользуемся сервисом <https://query1.finance.yahoo.com/>   
   История цен, сплитов и дивидендов  
   Постфикс для сервиса получения свечей выглядит так: /v8/finance/chart/GOOGL?symbol=GOOGL&period1=0&period2=9999999999&interval=3mo, где параметры

Интервалы:

* + &interval=3mo это 3 месяца.
  + &interval=1d это 1 день.
  + &interval=5m это 5 минут, возвращает 80 дней.
  + &interval=1m это 1 минута, возвращает 4-5 дней.
  + period1= unix timestamp представление даты, с которой вы хотите начать. Значения ниже начальной торговой даты будут округлены до начальной торговой даты.
  + period2= unix timestamp представление даты, на которой вы хотите закончить. Значения, превышающие последнюю торговую дату, будут округлены до последней доступной отметки времени.
  + Добавить данные до и после рынка: &includePrePost=true
  + Добавить дивиденды и сплиты: &events=div%2Csplit

Пример полного запроса:

*https://query1.finance.yahoo.com/v8/finance/chart/GOOGL?symbol=AAPL&period1=0&period2=9999999999&interval=1d&includePrePost=true&events=div%2Csplit*

Приведенный выше запрос вернет все данные о цене тикера GOOGL с интервалом в 1 день, включая данные до и после рынка, а также дивиденды и сплиты.

Для выполнения задания можно использовать любой интервал и период больше 15 дней от текущей даты.

В Loginom можно воспользоваться методом склейки Concat() компонента **Калькулятор(переменные)**, чтобы составить запрос на сервер.

Выходные данные должны представлять собой структурированный массив информации, с атрибутами:

Datetime (дата/время) – в формате Unix Timestamp, Open, High, Low, Close, Adj.Close, Volume, т.е. таблицу с данными колонками, полученными из json ответа от сервера. **(3 балла)**.