## Введение

Вашим первым видом автоматизированного тестирования является компонентное тестирование-тестирование отдельных сервисов по API. Для написания тестов вам понадобится уверенное знание структуры http-запросов и ответов, и различных инструментов, например, таких как postman (fiddler).

Для начала давай те посмотрим, что происходит под капотом системы. У нас есть фронт, в котором реализован пользовательский интерфейс и бэк, который прежде всего предназначен для сохранения состояния, например в бд.

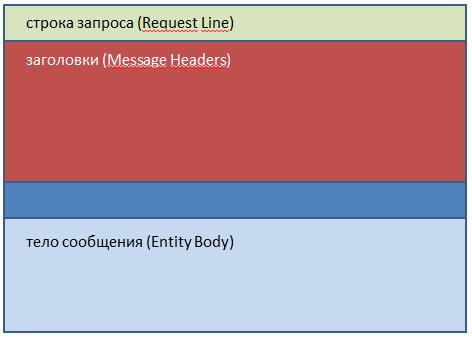
## Http - протокол

HTTP — широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам).   
Протокол HTTP предполагает использование клиент-серверной структуры передачи данных. Клиентское приложение формирует запрос и отправляет его на сервер, после чего серверное программное обеспечение обрабатывает данный запрос, формирует ответ и передаёт его обратно клиенту. После этого клиентское приложение может продолжить отправлять другие запросы, которые будут обработаны аналогичным образом. Задача, которая традиционно решается с помощью протокола HTTP — обмен данными между пользовательским приложением, осуществляющим доступ к веб-ресурсам (обычно это веб-браузер) и веб-сервером. На данный момент именно благодаря протоколу HTTP обеспечивается работа Всемирной паутины. Также HTTP часто используется как протокол передачи информации для других протоколов прикладного уровня, таких как SOAP, XML-RPC и WebDAV. В таком случае говорят, что протокол HTTP используется как «транспорт». API многих программных продуктов также подразумевает использование HTTP для передачи данных — сами данные при этом могут иметь любой формат, например, XML или JSON.

Само описание Http-протокола описано <https://developer.mozilla.org/ru/docs/Web/HTTP>

## Http-запрос

HTTP запрос состоит из трех основных частей, которые идут в нем именно в том порядке, который указан ниже. Между заголовками и телом сообщения находится пустая строка (в качестве разделителя), она представляет собой символ перевода строки.



1. строка запроса (Request Line)

2. заголовки (Message Headers)

Пустая строка (разделитель)

3. тело сообщения (Entity Body) – необязательный параметр

**Строка запроса** – указывает метод передачи, URL-адрес, к которому нужно обратиться и версию протокола HTTP.

**Заголовки** – описывают тело сообщений, передают различные параметры и др. сведения и информацию.

**тело сообщения**  - это сами данные, которые передаются в запросе.  Тело сообщения – это необязательный параметр и может отсутствовать.

## Строка запроса. Методы запроса

Основные:

* GET (запрашивает предоставление указанного ресурса, только получает данные)
* POST (предназначен для изменения данных на сервере, обычно используется для добавления нового ресурса)
* PUT (предназначен для изменения данных на сервере, обычно используется для изменения существующего ресурса, важно: должен присутствовать идентификатор ресурса)
* DELETE (предназначен для удаления ресурса)

## Строка запроса. URI

<https://example.com:5001/api/data/5?key1=val1&key2=val2>

1. Схема или протокол, https (также могут быть http, ftp, mailto и пр)
2. Имя хоста – example.com
3. Порт – 5001, по умолчанию для http – 80, https – 443
4. Путь запроса (path) - /api/data/5 – путь к физическому файлу на сервере или некий серверный идентификатор(псевдоним) ресурса.
5. Строка запроса (query string) – key1=val1&key2=val2. В качестве разделителя используется &. Строка запроса отделяется от остальной части «?»

Подробно здесь: <https://developer.mozilla.org/ru/docs/Web/HTTP/Basics_of_HTTP/Identifying_resources_on_the_Web_RU>

## Заголовки

Предназначены для взаимодействия между клиентов и серверов. Передаются раздельно на каждой строчке в виде «ключ:значение»

Например, при отправке данных на сервер используется атрибут Content-Type, чтобы сказать серверу в каком формате переданы данные в теле, например json

Content-Type: application/json

Соответственно, чтобы сказать серверу в каком формате клиент хочет получить данные, надо использовать атрибут Accept

Accept: application/xml

В данном случае сервер попробует передать данные в формате xml

Также важным атрибутом является авторизация клиента - Authorization

## Статус-коды

* Информационные 100 - 199
* Успешные 200 - 299
* Перенаправления 300 - 399
* Клиентские ошибки 400 - 499
* Серверные ошибки 500 - 599

## Передаваемые данные в теле запроса/ответа Json

Json – текстовый формат обмена данных, основанный на javascript.

Менее избыточный, чем xml

Синтаксис:

{

“ключ1“: значение1,

“ключ2”: значение2

}

Значением могут являться следующие типы:

* Строка (“…”)
* Число
* Логический тип (true|false)
* null – отсутствие значения
* Объект ({…})
* Массив ([…])

Например:

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*",

"mongo": {

"connectionString": "mongodb://localhost:27017/testing"

}

}

Соответственно здесь есть три ключа:

1. Logging (значением является объект, который содержит один ключ LogLevel, значений которого тоже является объект с 3 парами ключей изначений, где значения – строки)
2. AllowedHosts (значением является строка)
3. Mongo (значением является объект с одной парой ключ-значение)

Подробно тут: <https://developer.mozilla.org/ru/docs/Learn/JavaScript/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%8B/JSON>