**Методы**

1. **Метод GET**

**1.1. Получение информации от Сервера**

**1.2. Безопасный и Идемпотентный**

**1.3. Формат “Параметр = Значение” после знака ?**

**1.4. Кэшируемый**

**1.5. Небезопасный для клиента (т.к. может передавать конфиденциальную информацию в url)**

**1.6. Не имеет body (тела)**

**1.7. Запрос передаются в url (поэтому макс длина = 2048)**

**1.8. Успешный ответ Сервера на метод GET имеет body (тело)**

**1.9 GET запросы остаются в истории браузера = в закладках**

**2.0. GET запросы используются для извлечения данных**

**2.1. GET запросы сохраняются в логах сервера**

1. **Метод POST**

**2.1. Данные передаются в теле запроса (body). Ответ от Сервера также имеет тело.**

**2.2. Небезопасный (меняет состояние сервера)**

**2.3. Неидемпотентный (при одном и том же req разные resp)**

**2.4. Некэшируемый**

**2.5. Безопасный для клиента (т.к. инфа передается в теле)**

**2.6. Предназначен для передачи новой информации на Сервер**

**2.7. Отправляет текст/картинки**

**2.8. Длина запроса не ограничена**

1. **Метод PUT**

**3.2. Вся информация в теле запроса**

**3.3. Метод для замены существующего ресурса, используя данные представленные в теле запроса, но также может создавать новый ресурс(добавление файла например)**

**3.4. Успешный ответ не имеет тело.**

**3.5. Небезопасный метод, т.к. вносит изменения на сервер.**

**3.6. Идемпотентный метод.**

**3.7. Некэшируемый метод.**

**3.8. Длина запроса не ограничена.**

1. **Метод DELETE**

**4.1. Метод DELETE предназначен для удаления указанного ресурса.**

**Если метод DELETE успешно выполняется, то возможны следующие коды состояния ответа:**

**‒ 200 (OK). Удаление было выполнено, ответ содержит код и объект, описывающий состояние.**

**‒ 202 (Accepted). Удаление будет успешным, но ещё не выполнено.**

**‒ 204 (No Content) код ответа. Удаление было выполнено, но тело ответа отсутствует.**

**4.2. Небезопасный**

**4.3. Идемпотентный**

**4.4. Некэшируемый**

**4.5. Запрос и успешный ответ могут иметь тело**

1. **Метод Update**
2. **Данный метод обновляет данные на сервере (ключ или элемент)**
3. **PATCH частично обновляет и модифицирует**
4. **PUT обновляет и перезаписывает (полная замена)**

**6. Метод Head**

**6.1. Получение информации от Сервера**

**6.2. Но в ответе от Сервера приходят только заголовки (без тела)**

**6.3. Такой запрос может быть выполнен перед загрузкой большого ресурса (для экономии пропускной способности)**

**6.4. Аналог GET. Но запрашивает только заголовки (не содержит тело).**

**7. Метод Options**

**7.1. Описывает параметры соединения с Сервером**

**7.2. Запрос не имеет тела**

**7.3. Идемпотентный**

**7.4. Безопасный**

**7.5. НЕ кэшируется**

**7.6. Не производит никаких действий с Сервером**

**8. Отличия POST & PUT**

**8.1. POST идемпотентный. PUT не идемпотентный**

**8.2. POST создает новое. PUT меняет старое**

**8.3. POST отправляет данные на Сервер. PUT создает ресурс.**

**8.4. POST имеет тело запроса. PUT не имеет тела запроса в ответе от сервера.**

**9. HTTP- сессия состоит из трех фаз:**

**1. Клиент устанавливает соединение по транспортному протоколу TCP (или по другому протоколу, например UDP).**

**2. Клиент отправляет запрос на сервер и ждет ответ.**

**3. Сервер обрабатывает запрос и посылает ответ, в котором содержится статус-код и соответствующие данные.**

**Начиная с версии HTTP/1.1 2-я и 3-я фазы могут повторяться, т.е. клиенту позволено инициировать другой запрос.**

**10. URL-адрес**

**URL (Uniform Resource Locator) – унифицированный указатель ресурса в сети.**

**Адрес, по которому расположен ресурс.**

**Максим. длина = 2048**

**Части URL:**

URL состоит из различных частей, некоторые из которых являются обязательными, а некоторые - факультативными.

Например:

<http://www.example.com:80/path/myfile.html?key1=value1&key2=value2#Paragraf1>

1. Протокол - **http://**

2. Доменное имя - **www.example.com**

3. Порт - **:80**

4. Путь к файлу - **/path/myfile.html**

5. Параметры (ключ-значение) - **?key1=value1&key2=value2**

6. **Якорь** на другую часть того же ресурса - **#Paragraf1**

1) **Протокол** - http, https, ftp и т.д.

2) **Логин** - имя учетной записи пользователя для обращения к ресурсу

3) **Пароль** для входа в учетную запись

4) Хост или **Доменное имя** - помогает браузеру найти веб-сервер, где расположен ресурс, с помощью DNS. DNS переводит доменное имя в IP-адрес

5) **Порт** - Cтандартные значения (80 для http, 443 для https), прописывать в адресе сайта не обязательно. Если для работы ресурса нужен другой порт, его обязательно нужно указать.

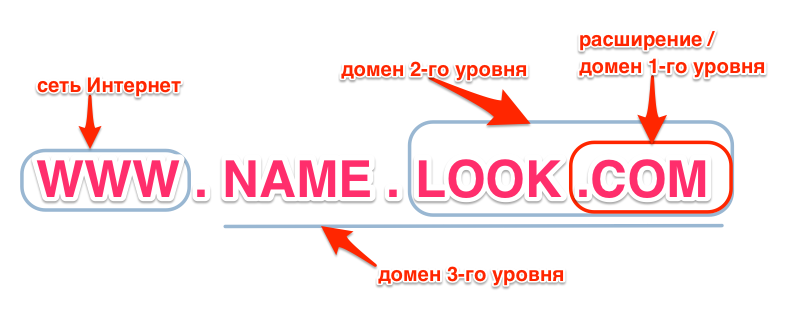
6) **Путь к странице** - часть URL-адреса, которая идет после домена

7) **Параметры** - это данные, которые браузер сообщает веб-серверу. Он использует их, чтобы выполнить дополнительные команды, перед тем как отдать содержимое ресурса. Они расположены после знака «?» и разделены символом «&»

8) **Якорь** - это ссылка на другую часть того же ресурса, например раздел статьи или видео. Находится якорь после символа **«#»**

**//Домены:**

1. До @ = имя почтового ящика
2. После @ = доменное имя
3. **.com** = доменная зона



**11. End-point API**

Конечная точка. Указывает точный адрес, по которому отправляется запрос.

Endpoint (конечная точка) представляет собой шлюз для взаимодействия серверных процессов приложения с внешним интерфейсом. Еndpoint – это кнопка на домофоне, где домофон – это API))

**12. Как записать параметры в URL HTTP запроса?**

Параметры HTTP-запроса прописываются в URL после знака ?. Между парами ключ-значение ставится знак & (амперсанд).

Например: ?name=Natasha&salary=1200.

**13. Какая разница между вэб сервис и вэб сервер**

**Веб-сервер** - это сервер реализующий http протокол.

**Веб-сервис** - это технология для взаимодействия между системами (между серверами).

**веб-сервисы** — это реализация абсолютно четких интерфейсов обмена данными между различными приложениями, которые написаны не только на разных языках, но и распределены на разных узлах сети.

**Веб-сервер** - это ПО, реализующее http протокол.

Общее у них только то, что любой веб-сервис не может работать без веб-сервера. А вот веб-сервер без веб-сервиса работает без проблем.