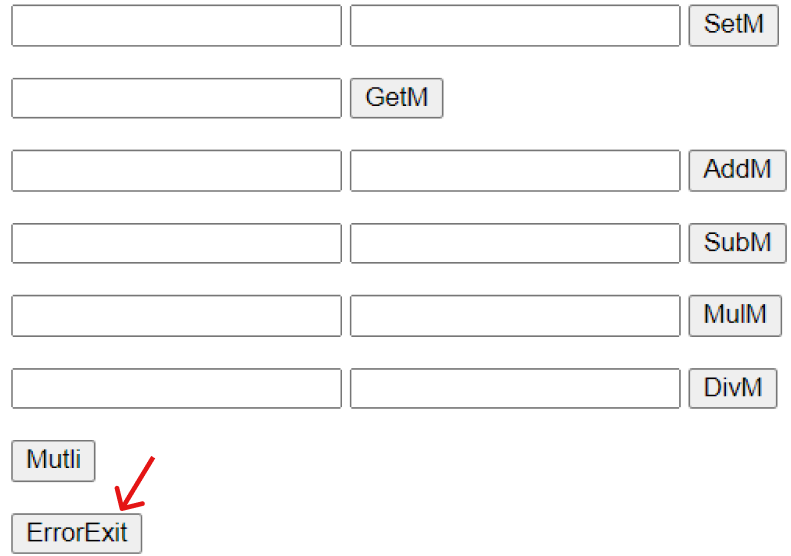
ErrorExit

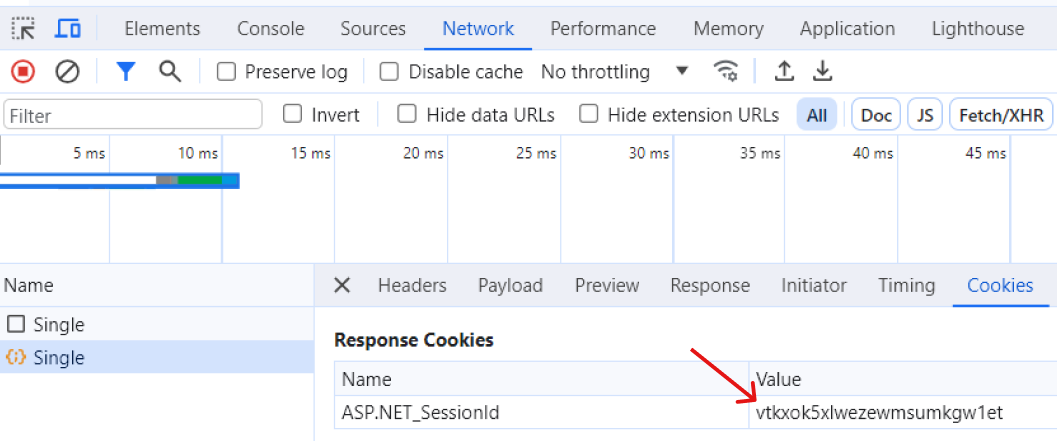
Если предыдущий запрос завершился, то оставшиеся вызовы игнорируются, а все данные в ВПС (внутренней памяти сервиса?) очищаются.

То есть, при нажатии на форме кнопки ErrorExit вызывается данный метод, после его вызова сервис больше не будет обрабатывать запросы в рамках данной сессии, а будет возвращать ошибку "Internal JSON-RPC error".

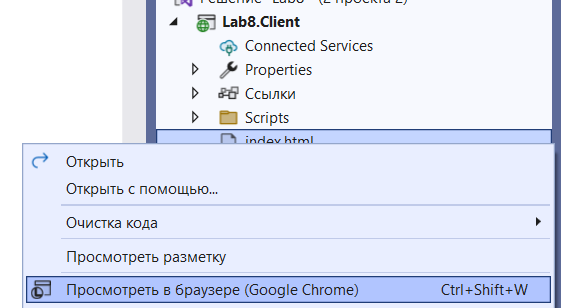


Если мы запустим сервис, затем клиента и у него вызовем данный метод, то, для создания новой сессии потребуется ***очистить историю браузера***.

Какая сейчас сессия можно отследить в браузере (до очистки истории и после значение id сессии поменяется):



Запуск клиента



JSON-RPC

- протокол удаленного вызова процедур, использующий формат JSON для передачи сообщений;

- последняя версия 2.0;

- транспорт: HTTP/HTTPS или TCP Socket.

**Объект запроса:**

- jsonrpc: строка, указывающая версию JSON-RPC протокола. Должна быть 2.0;

- method: строка, содержащая метод, который должен быть исполнен;

- params: параметры метода. Не обязательный;

- id: идентификатор установленный клиентом. Может содержать строку, число или null. Если не включен, то предполагается запрос без ответа (уведомление).

**Объект ответа:**

- jsonrpc: строка, указывающая версию JSON-RPC протокола. Должна быть 2.0;

- result: содержится в случае успешного выполнения. Содержит результат выполненного метода;

- error: содержится в случае ошибки (см. объект ошибки);

- id: обязательный параметр. Должен быть тот же, что и в объекте запроса.

**Объект ошибки:**

- code: код ошибки;

- message: описание ошибки;

- data: дополнительная информация об ошибке. Необязательный.

Пример rpc с позиционными параметрами:

****

****

Пример rpc с именованными параметрами:

****

****

**Пример rpc с уведомлением (запрос без ответа):**





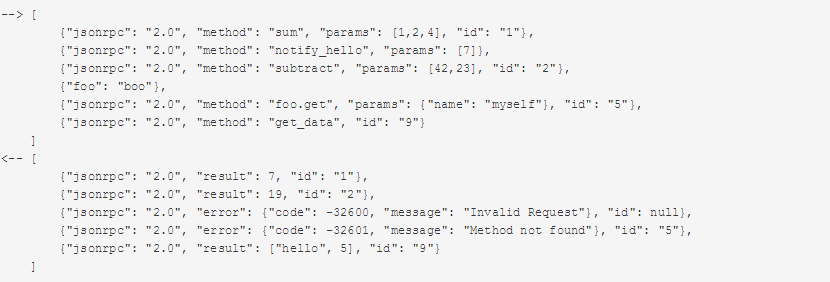
Примеры rpc с ответами-ошибками:

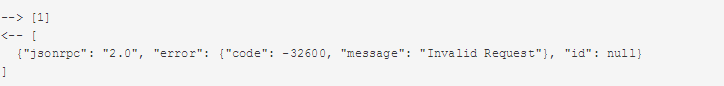


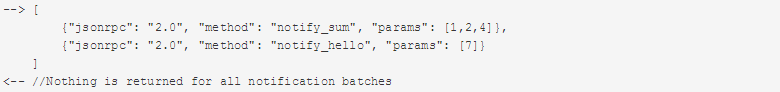


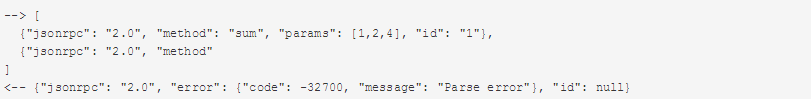


Примеры rpc с пакетом запросов:

****

****

****

****

**Коды ошибок:**

Коды от -32768 до -32000 зарезервированы для предопределенных ошибок. Коды, не описанные ниже, но входящие в данный интервал нужны для будущего использования:

-32700 - ошибка парсинга

-32600 - неверный запрос

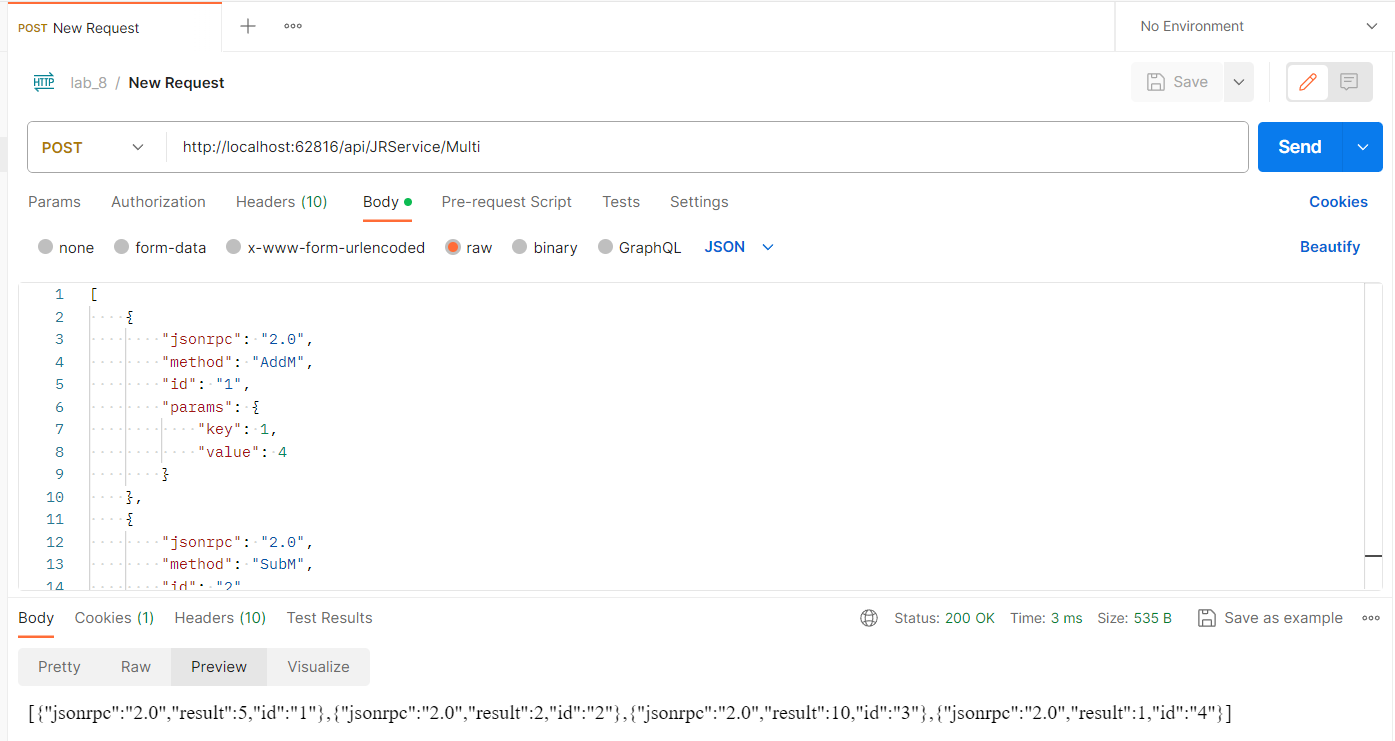
-32601 - метод не найден

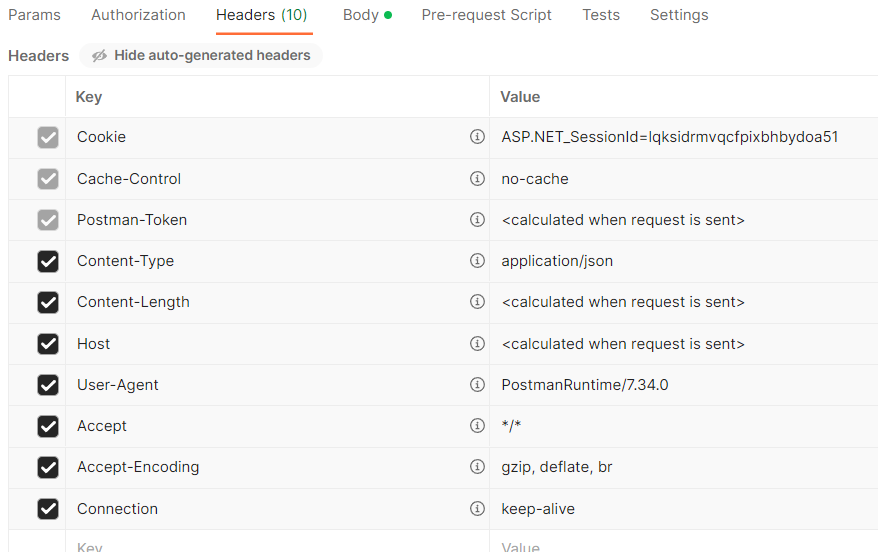
-32602 - неверные параметры

-32603 - внутренняя ошибка JSON-RPC

от -32000 до -32099 - ошибка сервера

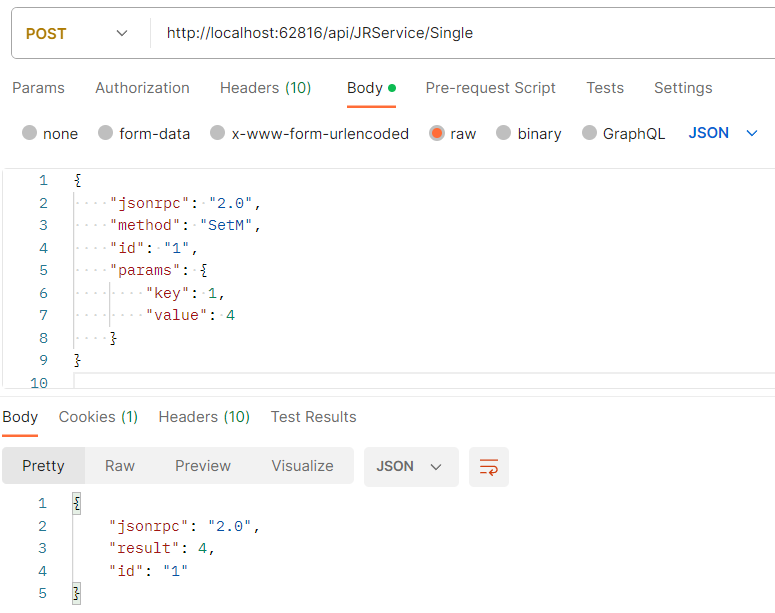
Postman



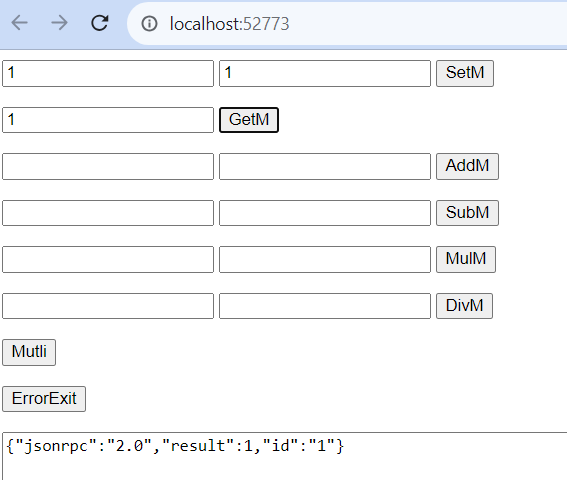


Не забываем, что значение x под ключом k хранится в рамках сессии, так что до оправки пакета запросов с действиями над значением x (кроме его установки) нужно:

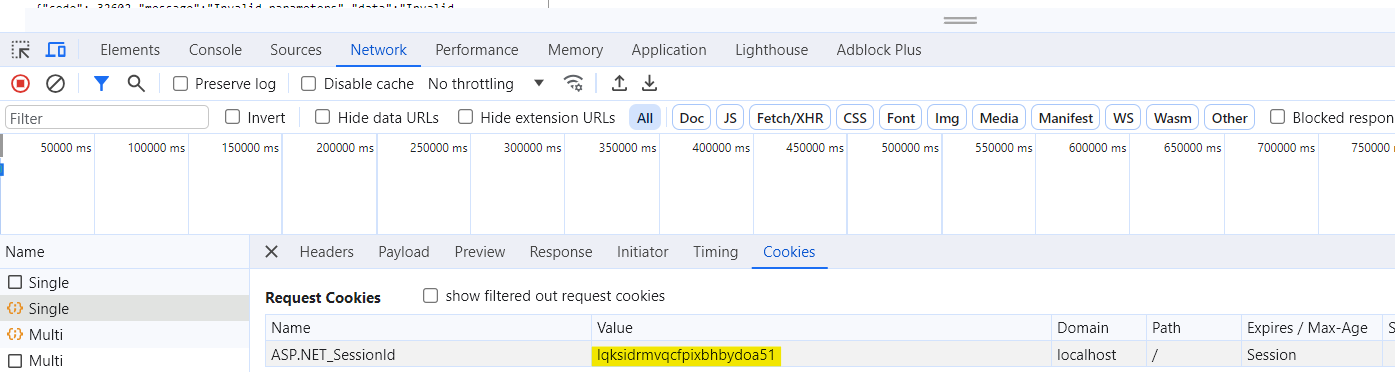
1. Либо вызвать SetM в postman:



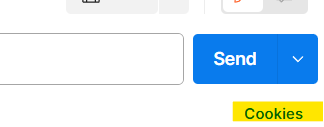
2. Либо вызвать SetM на страничке браузера:



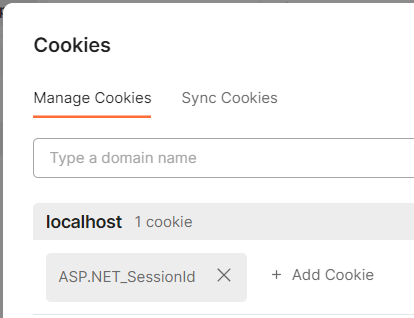
А затем скопировать значение куки сессии в браузере:



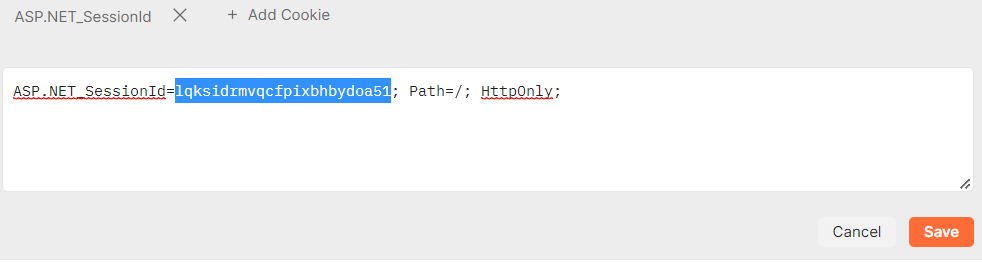
И добавить в postman, но перед этим в postman нужно выполнить любой запрос (он вернет jsonrpc с ошибкой), чтобы появилась сессия:



Нажимаем:



Меняем и нажимаем Save:



Что такое сессия и куки

*Сеанс = сессия.*

**Сеанс** – идентифицируемая серия запросов-ответов.

Первый запрос (с него создаётся сессия):

° Не содержит заголовок куки.

° Или куки есть, но в нем нет правильно заданного ид сессии.

° Или куки с ид есть, тогда сервер ищет если ли у него объект с таким ид (сессия), если не находит – это первый запрос.

В ответе на первый запрос сервер прописывает заголовок *set-cookie* в ответе, который должен обработать клиент. Сервер говорит о том, что начал новую сессию и теперь клиент должен сопровождать все свои ответы заголовком *cookie* с этим куки (содержит ид сессии).

timeout – параметр сервера (время ожидания между запросами).

*Задачи сеанса:*

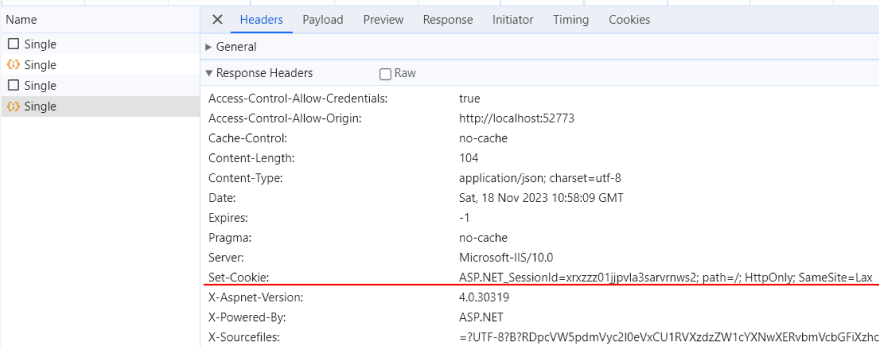
Сессия идентифицирует серию запросов-ответов. В рамках сеанса (сессии) сервер запоминает контекст. Сеансовый ключ (для шифрования) устанавливается в рамках одного сеанса.

**Cookie Vs. сессия**

|  |  |
| --- | --- |
| Cookies | Сессия |
| * Cookies – это файлы на стороне клиента, которые содержат информацию о пользователе | * Сеансы – это файлы на стороне сервера, которые содержат информацию о пользователе |
| * Cookie-файл заканчивается в зависимости от срока, установленного для него | * Сессия заканчивается, когда пользователь закрывает свой браузер или очищает кеш. |
| * Официальный (согласно стандарту RFC) максимальный размер файла cookie составляет 4 КБ. | * Размер данных сессии может зависеть от конфигурации сервера и используемого механизма управления сессиями. |
| * Файл cookie не зависит от сеанса | * Сеанс зависит от Cookie |

Сессии и куки в лабораторной

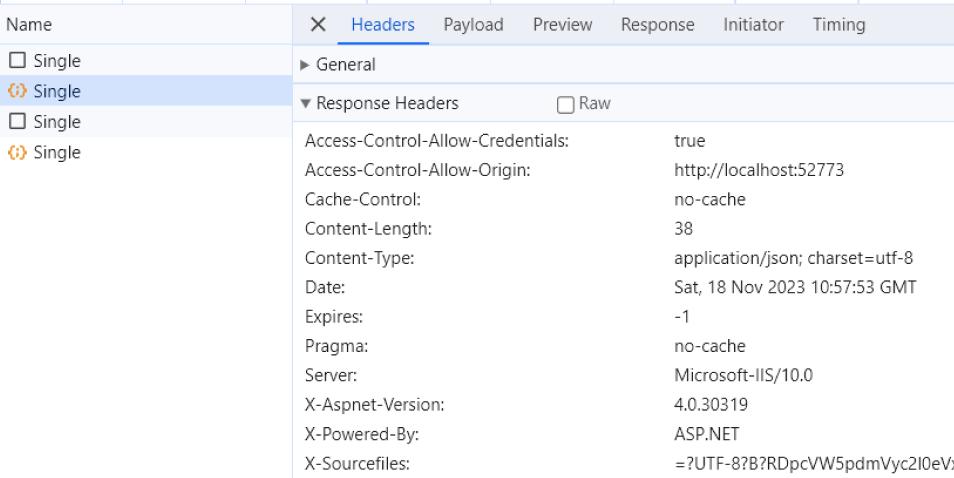
В заголовках ответа от сервера установлен заголовок Set-Cookies со значением, которое должен сохранить клиент:



Строка установки выглядит так:

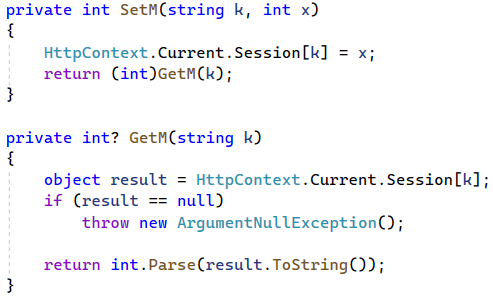
Set-Cookie: ASP.NET\_SessionId=xrxzzz01jjpvla3sarvrnws2; path=/; HttpOnly; SameSite=Lax.

Можно заметить, что это уже заголовок установлен лишь в ответе на 2-ой запрос, а не на 1-ый:



Это произошло из-за того, что браузеры внедряют политику *SameSite* для куки, которая определяет, в каких случаях браузер должен отправлять куки.

В ответе на 2-ой запрос видно значение данного атрибута: SameSite=Lax. Это значение говорит, что куки будут отправляться в ответе на запросы, вызванные непосредственно пользователем (при нажатии на кнопку, ссылку и т.д). Первым же запросом был вызван метод SetM, который из кода вызывает GetM и уже GetM возвращает ответ. То есть, вызов кода произошёл не при клике пользователя, а из кода. Второй же запрос вызвал непосредственно GetM, что и позволило серверу установить заголовок Set-Cookie.



Если отправить 3-ий запрос, то клиент добавит сохраненный куки в заголовок запроса Cookie:

