***Компоненты нагрузочных тестов в JMeter***

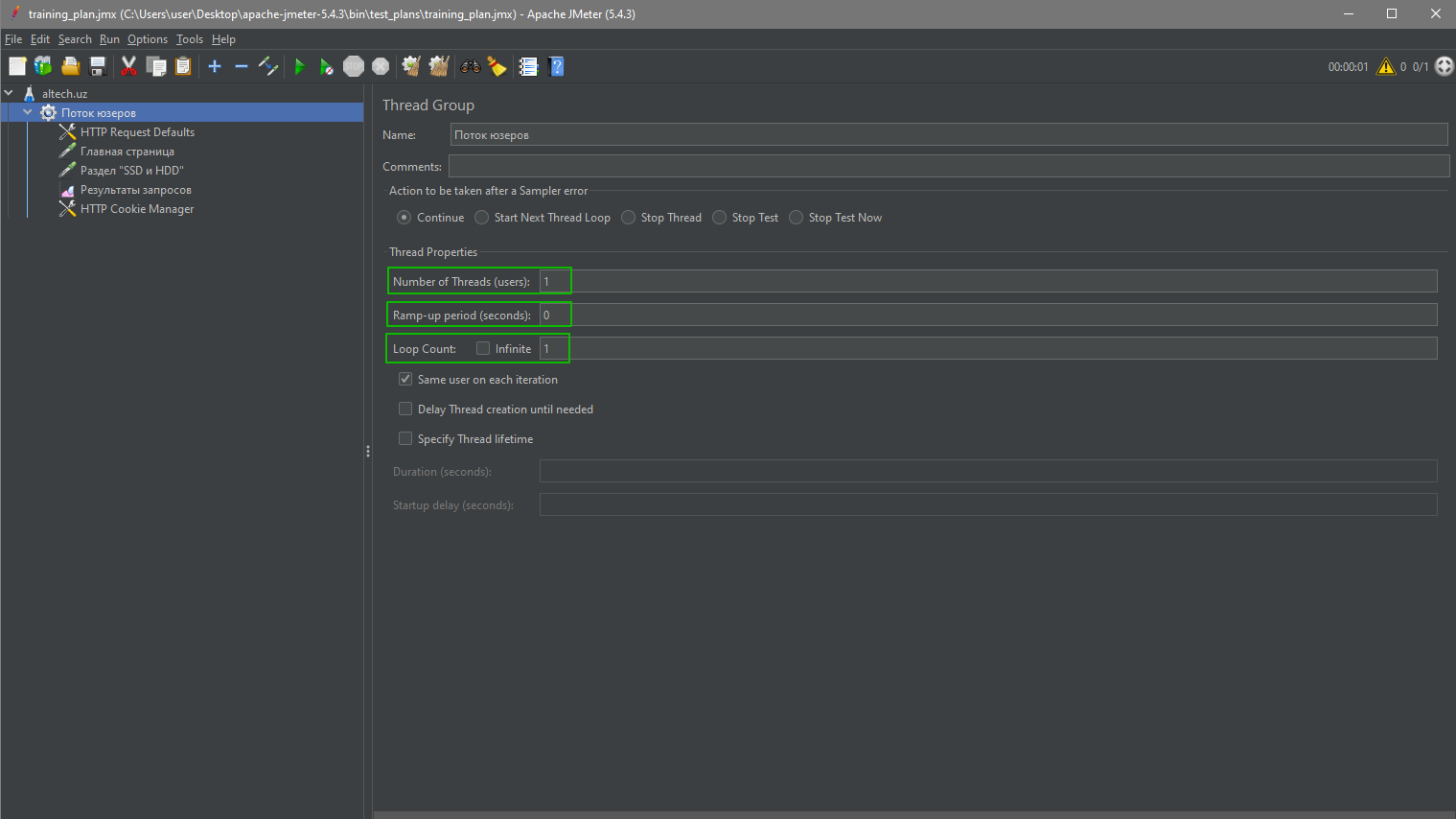
***Threads (Users) (Потоки)***

***Thread Group***

**Кликаем ПКМ по Тест-плану > Add > Threads (users) > Thread Group**

Это наш будущий поток юзеров, который будет использоваться при запуске тестов (имитация реальных юзеров)

В настройках потока есть поля «Number of Threads (users)», «Ramp-up period (seconds)» и «Loop Count»:



Первое отвечает за количество юзеров, которые будут «наплывать» на каждый наш запрос

Второе - период, за который количество наплывающих юзеров достигнет числа, указанного в первом поле. Например, если у нас в «Number of Threads» 5 юзеров, а в «Ramp-up period» 1 секунда, то при запуске теста будет отправлено 5 запросов за 1 секунду. Если выставим 5 секунд, то будет отправлено 5 запросов за 5 секунд, т.е. 1 запрос за 1 секунду

Третье отвечает за кол-во повторений запросов. Напр., если в числе юзеров стоит 1, но в «Loop Count» стоит «3», то будет отправлено 3 запроса

***Sampler (Сэмлеры)***

HTTP Request

**Кликаем ПКМ по Потоку юзеров > Add > Sampler > HTTP Request**

Это наш HTTP-запрос

- в поле «Protocol [http]» можем указать протокол (http или https),

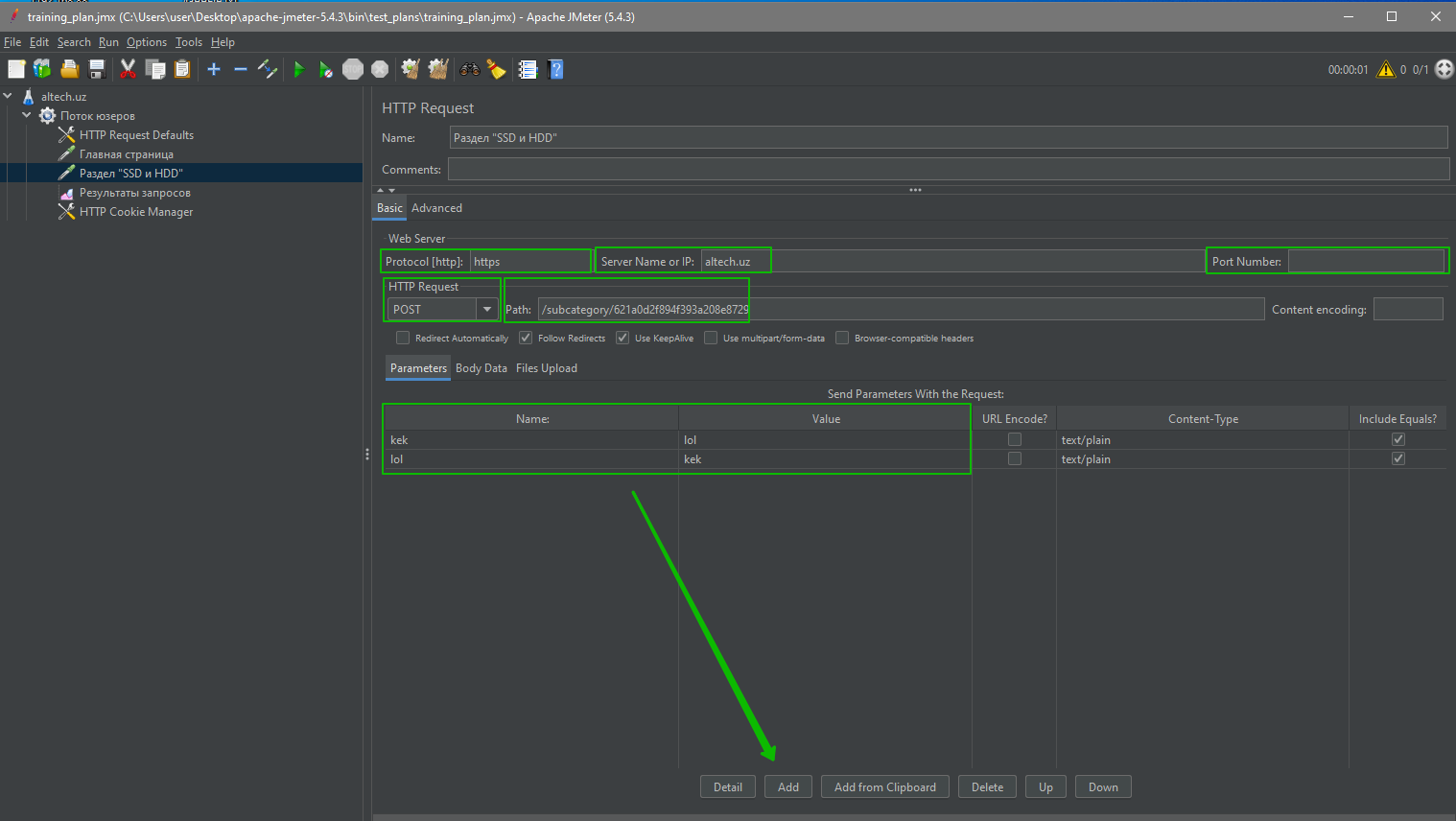
- в поле «Server Name or IP» указываем домен или IP целевого сайта,

- в поле «Port Number» можем указать номер порта целевого сайта,

- ниже в меню «HTTP Request» можем выбрать метод отправки запроса,

- в поле «Path» прописываем путь к HTML-файлу (какой-либо странице, если это не главная),

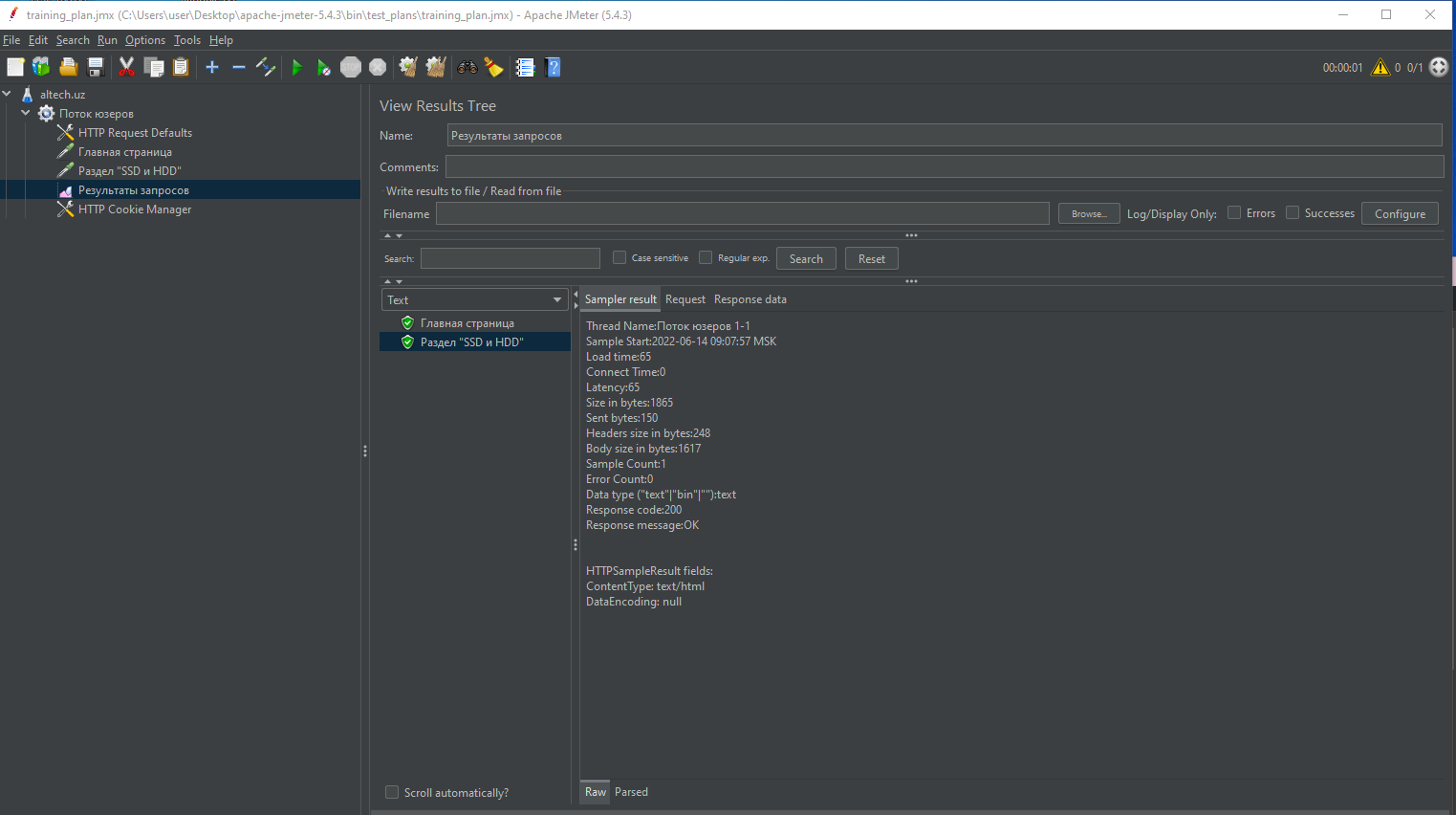
- во вкладке «Parameters» можем кликнуть кнопку «Add» внизу и добавить параметры запроса (в случае, если у нас POST- и другие подобные запросы)



***Listener (Слушатели)***

View Results Tree

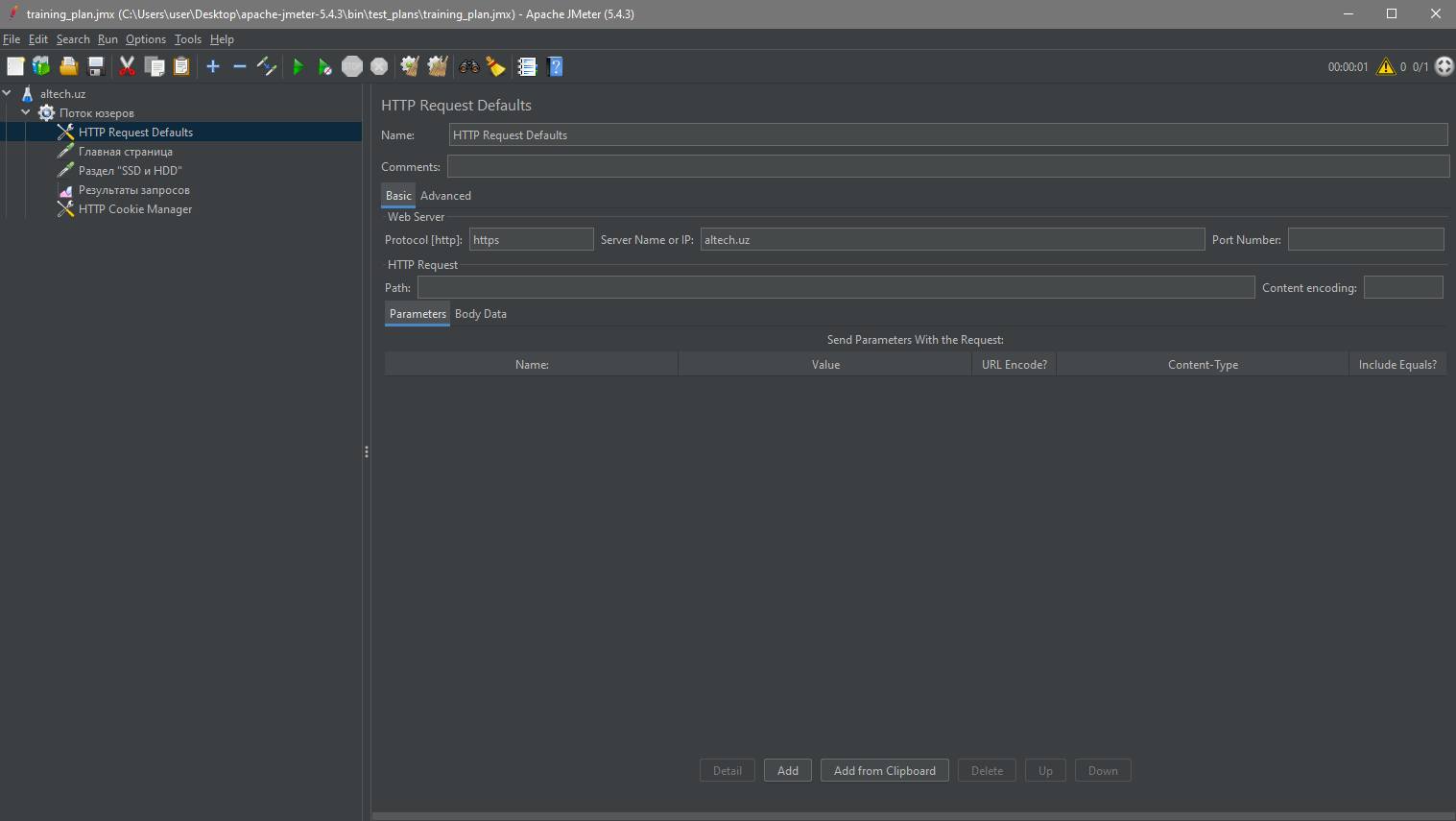
View Results Tree показывает результаты отправки запросов. Ничего сложного



***Config Element (Конфигурационные элементы)***

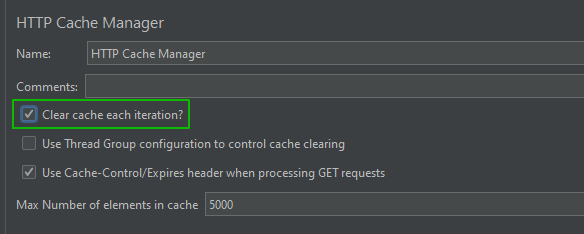
HTTP Request Defaults

Request Defaults - это что-то вроде дефолтной настройки, по которой будут отправляться наши запросы. Например, если в Defaults выставить параметры «Protocol [http]» и «Server Name or IP», а в самих запросах их удалить, то запросы будут отправлены именно по прописанным в Defaults параметрам. Довольно удобно!



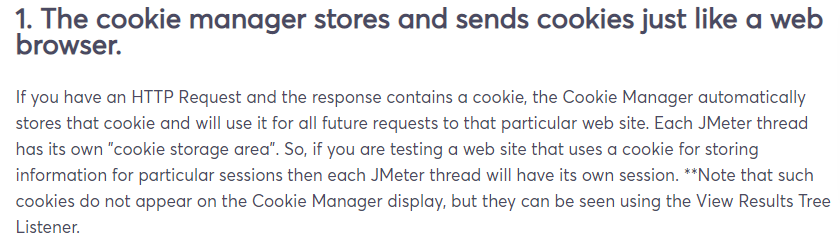
HTTP Cache Manager

Cache Manager позволяет симулировать кэширование браузера для большей реалистичности тестов. Для того, чтобы сайт думал, что все наши юзеры разные, нужно поставить галочку «Clear cache each iteration?», которая сбрасывает кэш при каждой отправке запросов



HTTP Cookie Manager

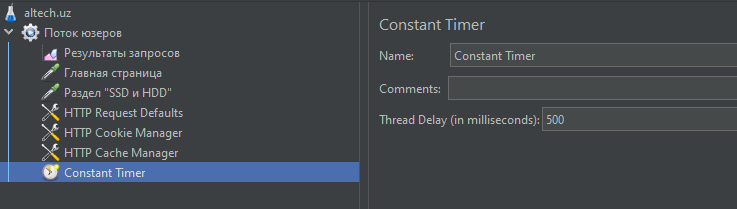
Cookie Manager сохраняет и посылает куки как браузер



***Timer (Таймеры)***

Constant Timer

Constant Timer используется для выставления времени между отправкой запросов в миллисекундах для большей реалистичности. Без таймера они отправляются одновременно



***Assertion (Проверки)***

Ассершены служат для проверки соответствия текущего результата ожидаемому результату. Если результаты тестов не совпадают, то мы будем видеть кастомное сообщение при кастомно составленных условиях проверки

Response Assertion

Response Assertion - проверка на соответствие ответа серверу заданному ответу, который мы ждём. Можем указать соответствие ответа по:

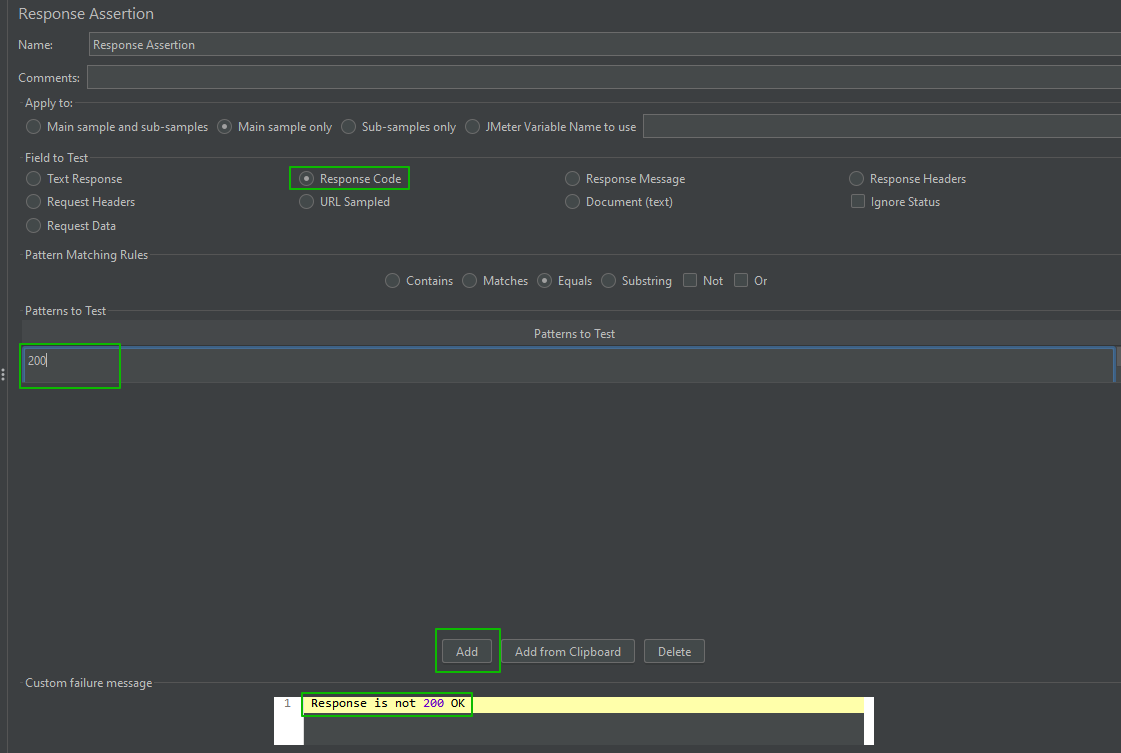
- коду ответа

- тексту ответа

- заголовкам ответа

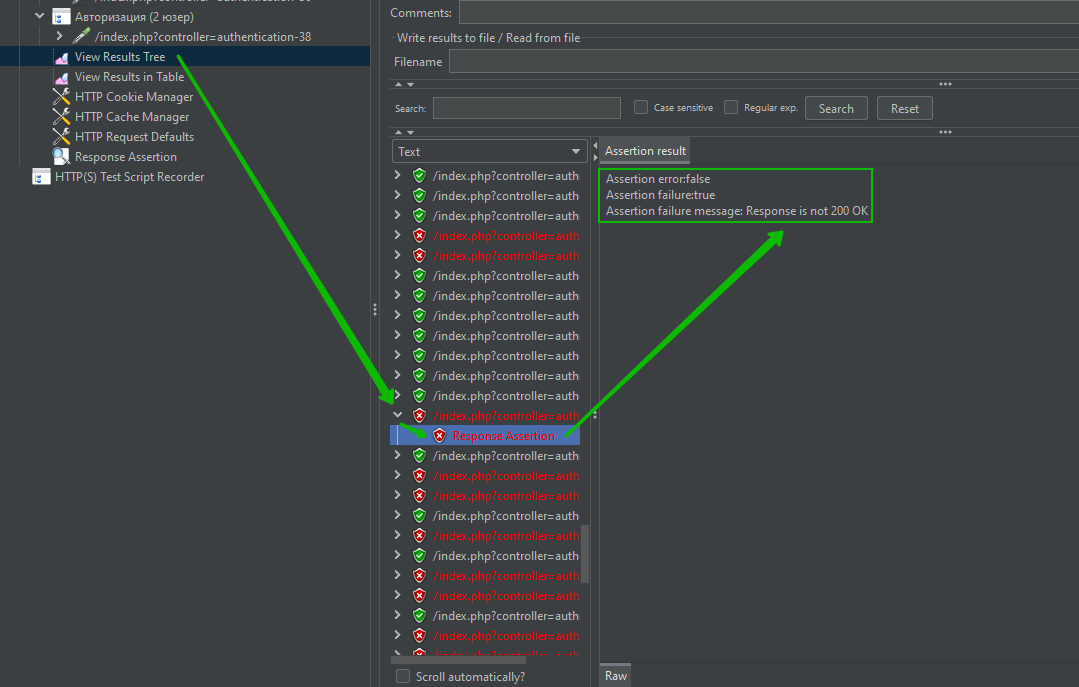
То же самое можно сделать и для запросов

Например, мы можем задать условие, при котором проверка будет проходить при коде ответа «200». Если мы получим другой код, то получим сообщение об ошибке, которое сами составим:



В «Field to Test» выберем **«Response Code»**, в «Patterns to Test» введём **«200»**, в «Custom failure message» введём **сообщение, которое будет показываться в случае, если мы НЕ получим ответ 200.**

Запустим тесты и посмотрим зафейленные запросы:



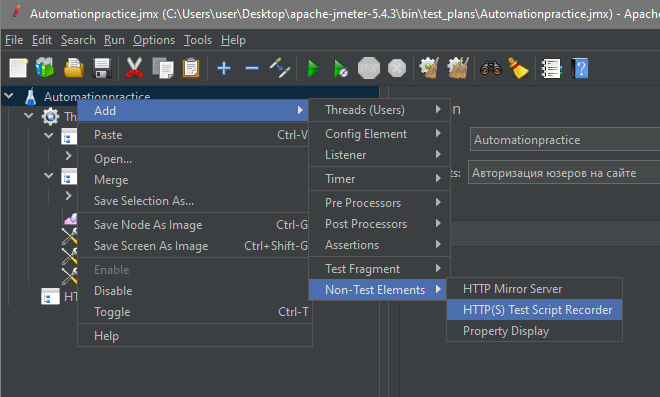
***Запись HTTP(S) скриптов***

Хороший видос от индуса: <https://www.youtube.com/watch?v=voYj16cETAM&ab_channel=AutomationStepbyStep>

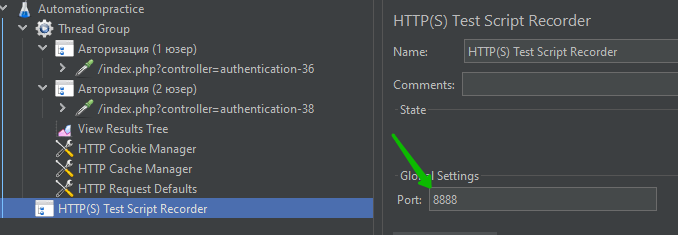
Запись скриптов позволяет выявить определённые HTTP-запросы, которые понадобятся нам в дальнейших тестах. Например, можно записать процесс авторизации на сайте и увидеть HTTP-запросы с данными для логина, а затем тестить авторизацию. Ну и прочая шляпа.

***Подготовка окружения для записи в JMeter***

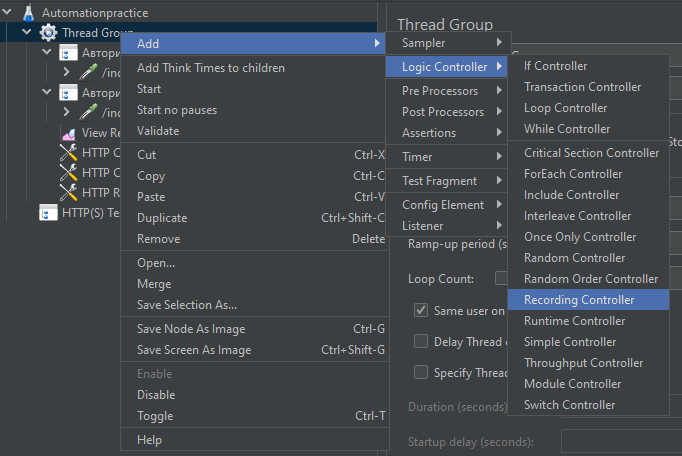
Сначала в нашем тест-плане создадим **Thread Group**. Затем снова кликаем ПКМ на **тест-план > Add > Non-Test Elements > HTTP(S) Test Script Recorder:**



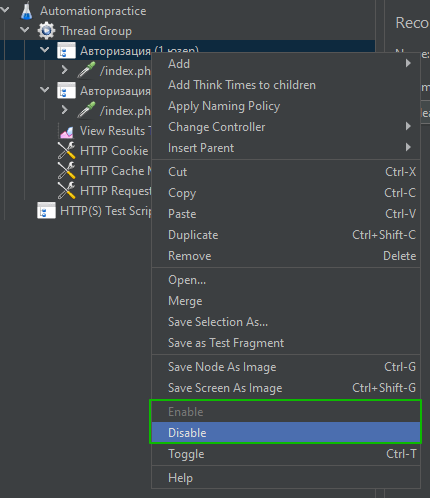
В рекордере по дефолту стоит порт 8888, но его можно сменить по усмотрению на любой другой. В дальнейшем мы будем указывать этот порт в настройках прокси:



Внутрь Thread Group закинем 1 (можно больше, в зависимости от нужд) **Recording Controller.** Для этого: **ПКМ по Thread Group > Add > Logic Controller > Recording Controller:**



Таким образом**, HTTP(S) Test Script Recorder** записывает скрипты и автоматом закидывает их в наши **Recording Controller**’ы. Recording Controller’ы можно размножить и сделать каждый для определённых сценариев записи. Их можно включать и выключать (и не только их):

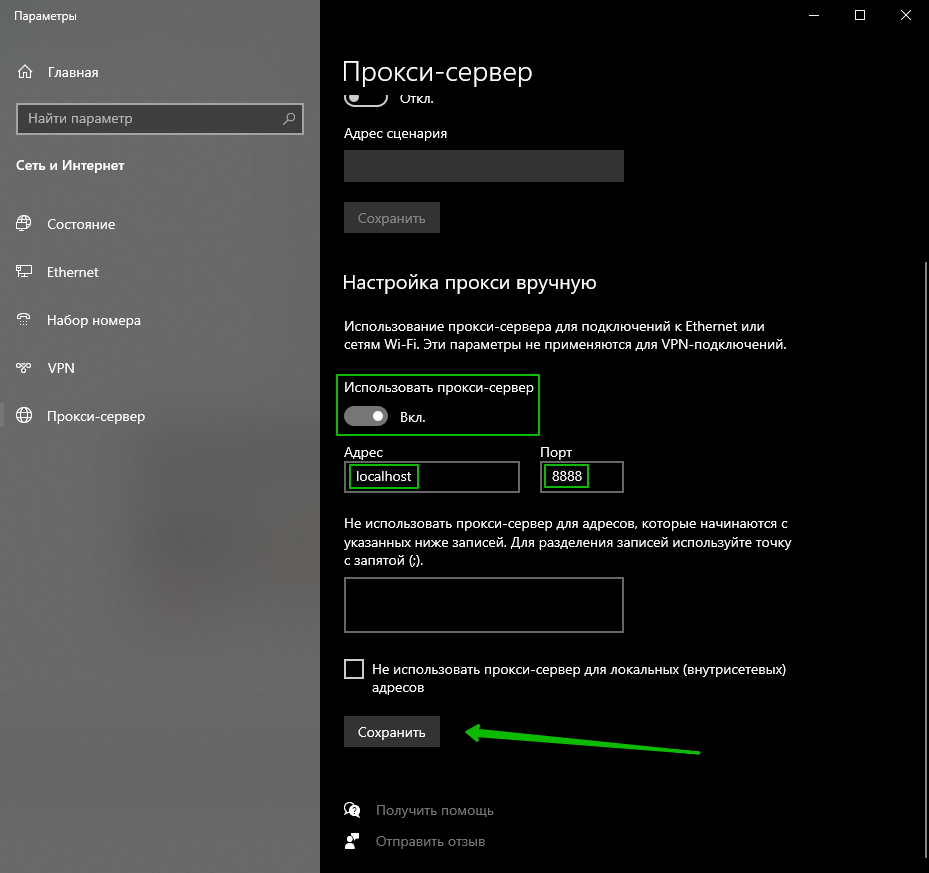


Не забываем ещё добавить HTTP Cookie Manager, HTTP Cache Manager и HTTP Request Defaults (по усмотрению) для большей реалистичности теста, а также View Results Tree для просмотра результатов отправки запросов. Готово!

***Настройка прокси для записи скриптов***

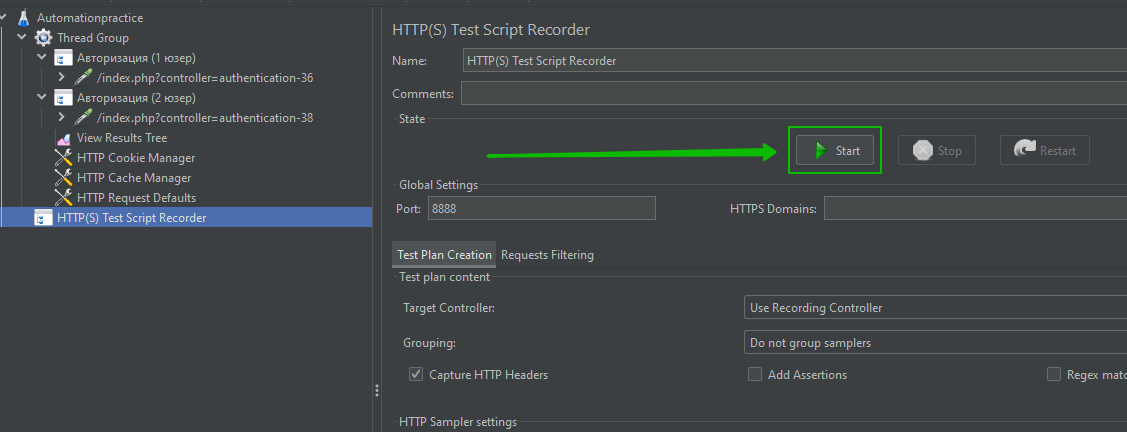
Наш скрипт рекордер будет выступать в качестве прокси, т.е. стоять между клиентом и сервером. Поэтому нам нужно зайти в настройки прокси и кое-што там сделать...

На Windows 10 это око выглядит так:



Включаем прокси, в адресе ставим «localhost», в порте - порт, указанный в нашем рекордере (в данном случае «8888») и сохраняем настройки. ***Теперь сайты будут открываться ТОЛЬКО после включения рекордера.***

Всё готово для записи. Для этого заходим в наш рекордер и включаем запись:



Теперь при посещении сайтов в Recording Controller’ах начнут появляться записанные запросы.