**Основные понятия нагрузочного тестирования**

**Нагрузочное тестирование** (или **тестирование производительности**, Performance testing) — это автоматизированное тестирование, имитирующее работу определенного количества пользователей в приложении. Нагрузочное тестирование имеет целью выяснить граничные условия нагрузки на систему, при которой она продолжает работать стабильно и с приемлемым временем отклика, а также оценить способность системы правильно функционировать в случае превышении планируемых нагрузок.

Более конкретно выделяют следующие **виды нагрузочного тестирования:**

* **Тестирование производительности** – измеряется скорость работы системы при идеальных условиях и максимальной нагрузке.
* **Нагрузочное тестирование** – это те же тесты производительности, но в которых система подвергается различным нагрузкам.
* **Объемное тестирование** – приложение нагружается большим количеством данных, чтобы определить, когда достигаются условия, при которых система перестает работать.
* **Стресс тестирование** – поведение системы при недостатке ресурсов (ресурсов процессора, дискового пространства, обрывов сети и т.п.).

**План нагрузочного тестирования**

1. Определение целей тестирования, разработка профилей нагрузки.
2. Настройка и отладка нагрузочных тестов в JMeter.
3. Многократное воспроизведение нагрузочных тестов в соответствии с профилями нагрузки.
4. Анализ результатов и формирование отчёта по разработанному шаблону**.**

**JMeter – Login**

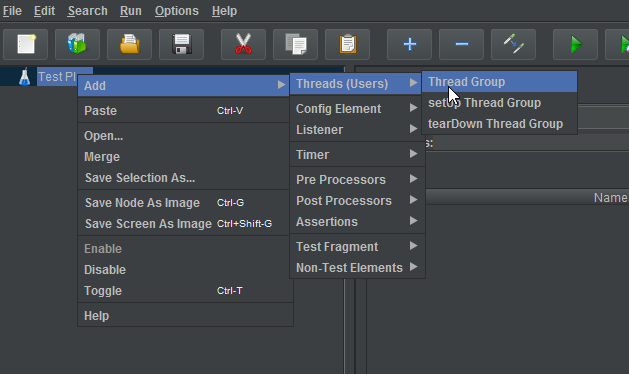
Для любого программного приложения, предназначенного для массового обслуживания пользователей, необходимо проводить нагрузочное тестирование на предмет его надежности и отказоустойчивости. А так как любой web-сайт — это по своей сути система массового обслуживания, то проверка его на отказоустойчивость всегда является неотъемлемой частью разработки. Существуют различные решения для проведения нагрузочного тестирования веб-приложений, я хочу рассказать об использовании такого средства, как Apache JMeter.

JMeter является очень мощным инструментом нагрузочного тестирования с возможностью создания большого количества запросов одновременно.

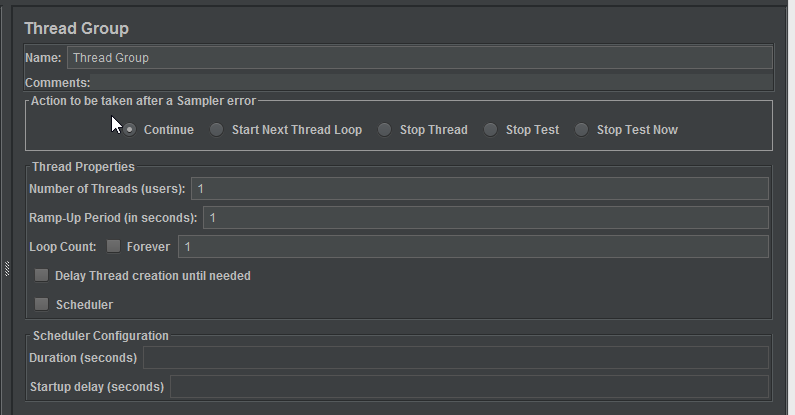
Метод «тыка» неприменим к JMeter. Поэтому если хотите его использовать, то необходимо изучать мануал. Я сейчас опишу создание простого нагрузочного теста с использованим данной тулы. В итоге мы получим тест-план с минимально необходимым набором модулей.

1. Запускаем JMeter.
2. В ветку Test Plan добавляем элемент **Thread Group** - отдельный тест-кейс, в который нужно записывать данные от прокси-сервера.

|  |
| --- |
| *Замечание:* *Прокси-сервер — промежуточный сервер (комплекс программ) в компьютерных сетях, выполняющий роль посредника между пользователем и целевым сервером, позволяющий клиентам как выполнять косвенные запросы (принимая и передавая их через прокси-сервер) к другим сетевым службам, так и получать ответы. Сначала клиент подключается к прокси-серверу и запрашивает какой-либо ресурс, расположенный на другом сервере. Затем прокси-сервер либо подключается к указанному серверу и получает ресурс у него, либо возвращает ресурс из собственного кэша (в случаях, если прокси имеет свой кэш). В некоторых случаях запрос клиента или ответ сервера может быть изменён прокси-сервером в определённых целях. Прокси-сервер позволяет защищать компьютер клиента от некоторых сетевых атак и помогает сохранять анонимность клиента.* |

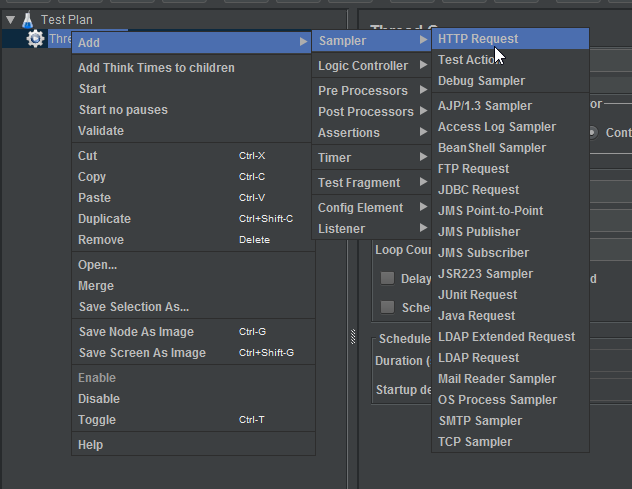


Элемент **Thread Group** позволяет задавать параметры генерируемой на приложение нагрузки. Основными его параметрами являются:

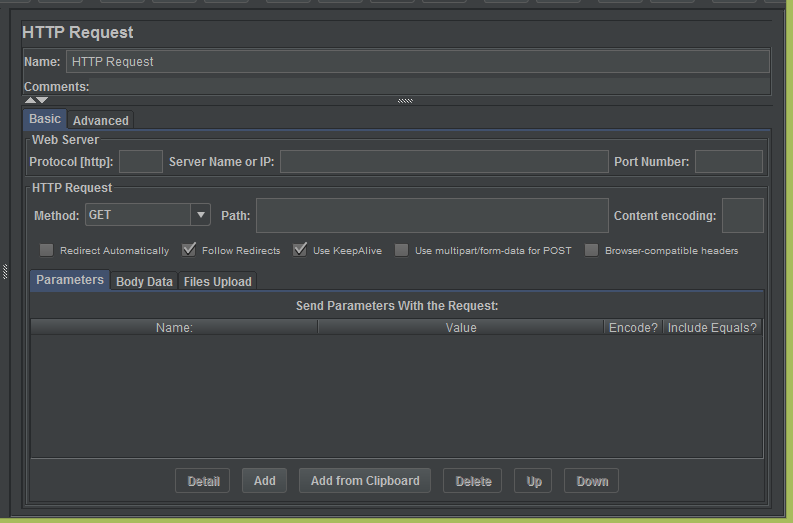


* **Number of threads** - количество имитируемых пользователей, одновременно работающих с сайтом;
* **Ramp-up period** - промежуток времени, через который выполняется запуск очередного процесса;
* **Loop count** - количество раз, которое будет выполняться сценарий внутри Thread Group;
* **Forever** - сценарий будет выполняться всегда, пока не будет прерван явно;
* **Scheduler** - планировщик времени работы сценария;
* **Action to be taken after a Sample Error** - действие, выполняемое в случае, если запрос вызовет ошибку.

1. Теперь нужно создать собственно сценарий теста, т.е. набор различных действий для создания нагрузки на сайт. Можно создавать сценарий вручную. Для этого нужно добавить N-ое количество элементов **HTTP Request**, которые добавляются так Thread Group -> Add -> Sampler -> HTTP Request.

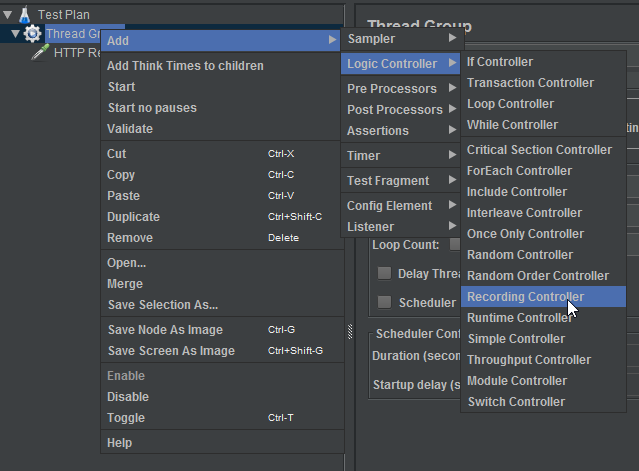


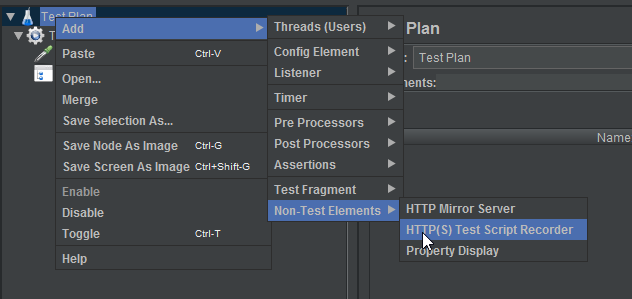
Появляется окно настроек, представленное на рисунке ниже. В соответствующие поля устанавливаем адрес сайта, порт (если нужно), путь к странице.



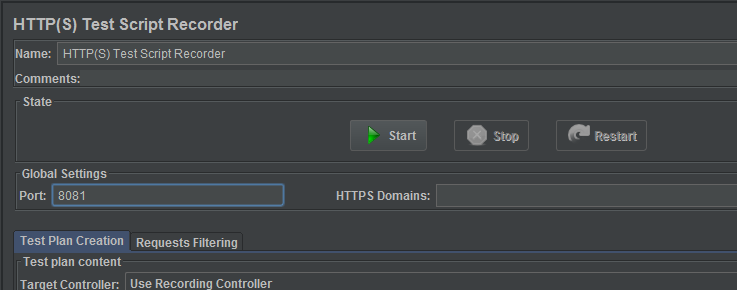
Кроме того, можно добавить параметры для запросов. Это можно сделать вручную, но это довольно утомительно. В JMeter есть такая возможность, которая называется **запись тестов через проксирование**. Т.е. мы будем выполнять любые действия через браузер, и при этом все необходимые элементы HTTP Request будут создаваться без нашего участия. Смотрим далее по пунктам, как это делается.

1. Добавляем **Recording Controller** (Thread Group -> Add -> Logic Controller -> Recording Controller). В данный элемент будут сохраняться все наши действия(requests), которые мы будет делать в браузере.

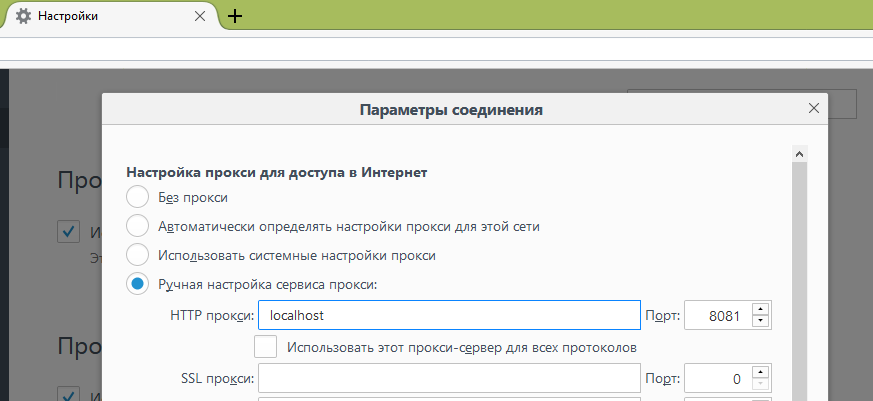


1. Добавляем элемент **HTTP(s)Test Script Recorder**. Он нам нужен только, чтобы записать запросы.  
     
   

Нужно изменить номер порта прокси-сервера, если порт по умолчанию 8080 у вас уже занят, например, можно поставить 8081.

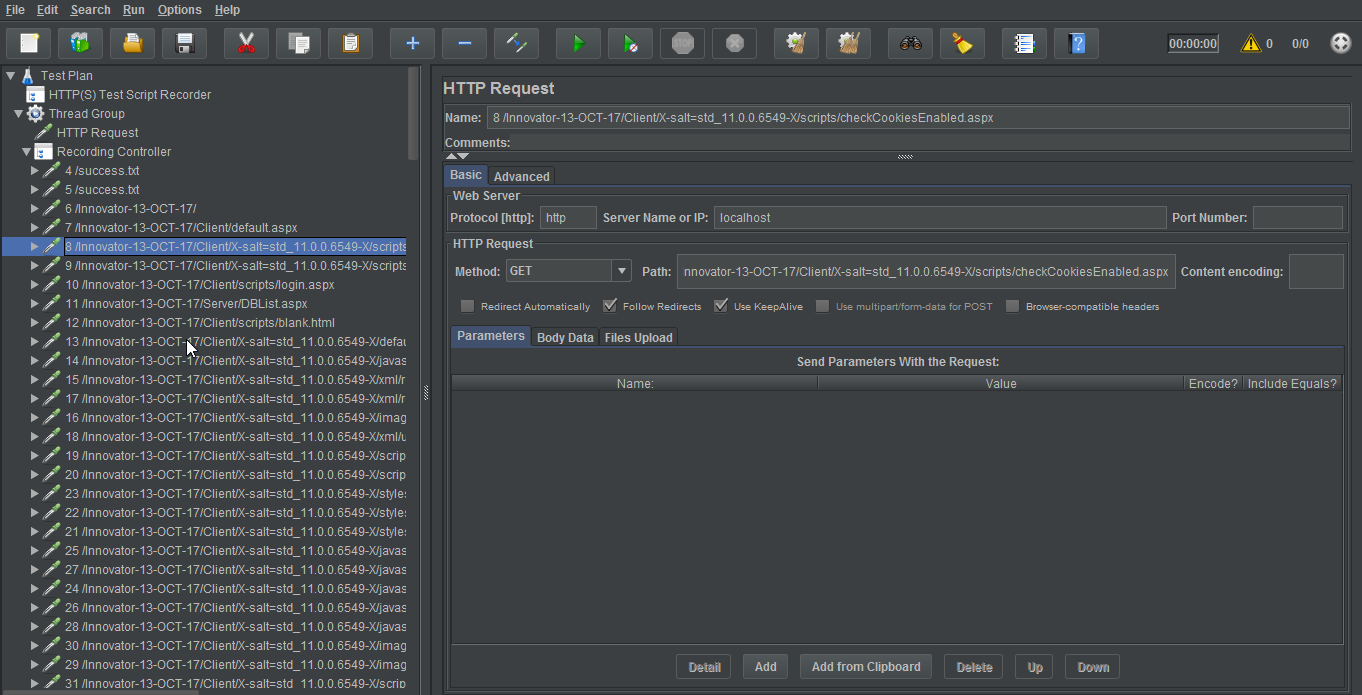


1. Далее необходимо изменить настройки браузера. Предпочтительно выбрать Mozilla Firefox. В настройках браузера нужно указать адрес прокси и порт, как на скриншоте ниже. Можно убедиться, что браузер входит в интернет именно через нее, для этого перейти на любой сайт в интернете, при этом страница не должна загрузиться.

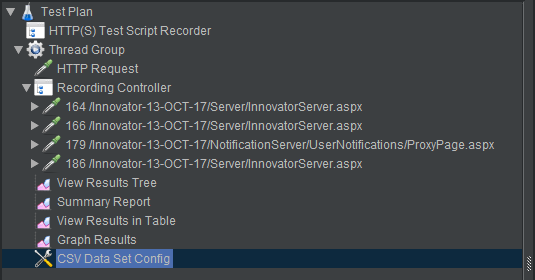


Для записи запросов необходимо нажать кнопку **Start** вокне настройки элемента **HTTP(s)Test Script Recorder**. Далее необходимо через браузер пройти тест-сценарий (открываем страницу тестируемого сайта, логинимся, выполняем различные действия, посещаем разных страницы) и нажать кнопку **Stop**. Сценарий сохранится в элемент **Recording Controller.**

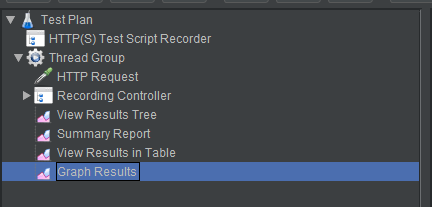
Получаем приблизительно следующее:



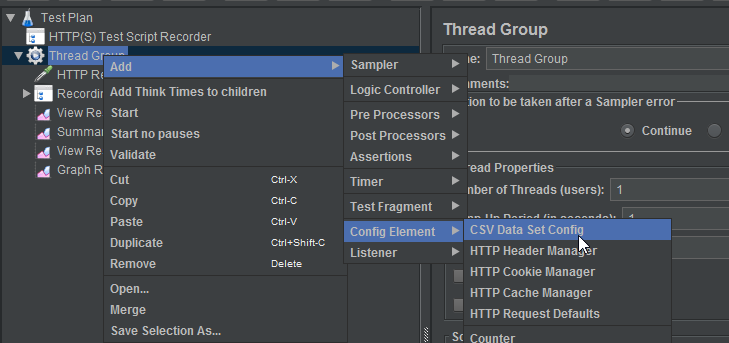
Необходимо отсортировать записанные реквесты, оставив только необходимые для логина и удалив остальные (.jpg,.png и т.п.).



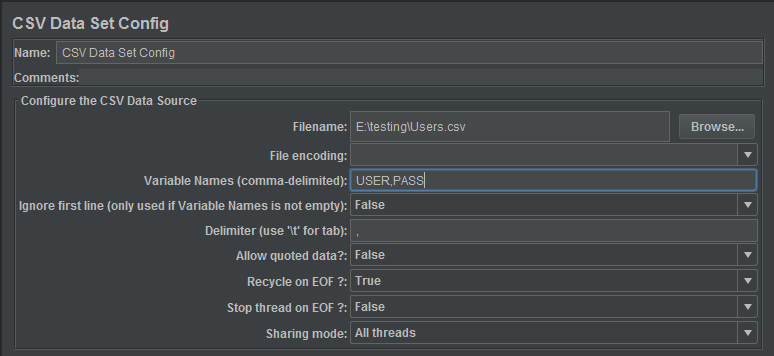
1. Чтобы наблюдать результаты тестов, а также следить за ходом выполнения, нужно в ветку Thread Group добавить несколько элементов мониторинга - слушателей (для визуализации и логирования результатов тестирования). Можно выбрать наиболее наглядный и удобный для конкретного случая элемент: **View Results Tree, View Results in Table, Graph Results, Summary Report.**



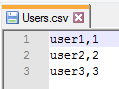
|  |
| --- |
| *Замечание:* *очень полезный элемент View Results Tree, в котором можно смотреть все параметры и содержимое запросов и ответов* |

1. Реализуем загрузку списка пользователей из файла для имитации того, что некоторое количество пользователей логинится на ресурс. Делается это с помощью добавления элемента CSV Data Set Config (Thread Group -> Add -> Config Element -> CSV Data Set Config).  
     
   

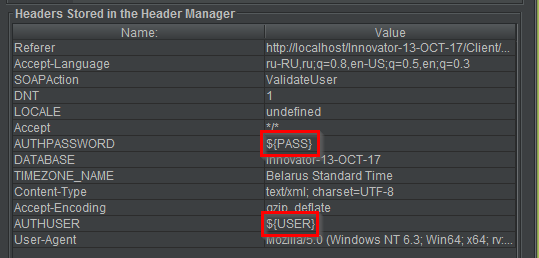
В появившемся окне указываем имя, путь к файлу, а также имена переменных. Эти имена вы задаете сами, они в дальнейшем будут использоваться при формировании запроса. К примеру, для связки юзер\пароль, два имени переменных: USER, PASS.

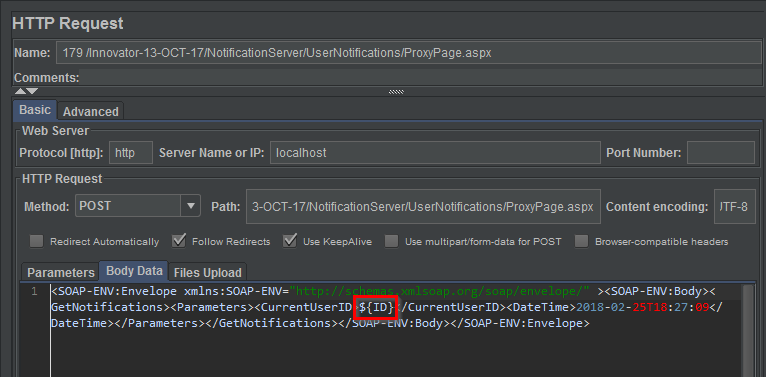


Сам файл с расширением csv может выглядеть так:

**

Пользователей мы подали на вход JMeter. Теперь нужно сказать ему, чтобы он их использовал там, где это требуется. Делается это так, ищем тот элемент HTTP Request, который отвечает за авторизацию. Как правило он в себе будет содержать параметры запроса, а именно: тот логин и пароль, которые вы ввели, записывая сценарий через браузер. Вместо конкретных значений имени пользователя и пароля подставляем имена наших переменных в таком формате ${USER}, ${PASS}. Это выглядит так:





После этого JMeter будет циклически использовать эти данные для разных потоков, обеспечивая таким образом вполне реалистичную нагрузку на сайт.

1. Теперь мы можем запустить Test Plan кликнув на  на тулбаре и увидеть результаты отработки в элементах Listeners.