МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«27» 12 2017 г.

АВТОМАТИЗАЦИЯ НАГРУЗОЧНОГО ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Методические указания по выполнению лабораторной работы по дисциплине "Тестирование программного обеспечения" для студентов направления подготовки 09.03.04 "Программная инженерия"

Курск 2017

УДК 004.65

Составители: В.Г. Белов, Т.М. Белова

Рецензент
Кандидат технических наук, доцент кафедры программной инженерии ЮЗГУ И.Н. Ефремова

Автоматизация нагрузочного тестирования программного обеспечения: методические указания по выполнению лабораторной работы по дисциплине "Тестирование программного обеспечения" для студентов направления подготовки 09.03.04 "Программная инженерия" / Юго-Зап. гос. ун-т; сост.: В.Г. Белов, Т.М. Белова, – Курск, 2017. – 26 с.: ил. 14.

Изложена последовательность действий с программным обеспечением JMeter для нагрузочного тестирования веб-приложений.

Материал предназначен для студентов направления подготовки бакалавров 09.03.04 «Программная инженерия», а также будет полезен студентам всех направлений подготовки, изучающим технологии нагрузочного тестирования.

Текст печатается в авторской редакции.

Подписано в печать 27.12.17. Формат 60x84 1/16.
Усл. печ. л. 1,3. Уч.-изд. л. 1,2 Тираж 100 экз. Заказ 4374. Бесплатно.
Юго-Западный государственный университет
305040, Курск, ул.50 лет Октября, 94.

Содержание

1 Цель лабораторной работы

Целью лабораторной работы является приобретение знаний, умений и навыков нагрузочного тестирования при помощи программы JMeter для разработки программно-информационных систем.

2 Порядок выполнения лабораторной работы

2.1 Необходимые теоретические сведения

Нагрузочное тестирование (load testing) — подвид тестирования производительности, сбор показателей и определение производительности и времени отклика программно-технической системы или устройства в ответ на внешний запрос с целью установления соответствия требованиям, предъявляемым к данной системе (устройству).

Термин нагрузочное тестирование может быть использован в различных значениях в профессиональной среде тестирования ПО. В общем случае он означает практику моделирования ожидаемого использования приложения с помощью эмуляции работы нескольких пользователей одновременно. Таким образом, подобное тестирование больше всего подходит для многопользовательских систем, чаще — использующих клиент-серверную архитектуру (например, веб-серверов). Однако и другие типы систем ПО могут быть протестированы подобным способом. Например, текстовый или графический редактор можно заставить прочесть очень большой документ; а финансовый пакет — сгенерировать отчёт на основе данных за несколько лет. Наиболее адекватно спроектированный нагрузочный тест даёт более точные результаты.

Задержка (latency) - это показатель того, насколько быстро сервер реагирует на запросы клиента. Измеряется в миллисекундах (мс). Задержкой часто называют временем отклика. Более низкие показатели

указывают на более быстрые ответы. Задержка измеряется на стороне клиента с момента отправки запроса до получения ответа. В это число включены сетевые накладные расходы.

Пропускная способность (throughput) - это количество запросов, которые сервер может обрабатывать в течение определенного промежутка времени, обычно отображается как запросы в секунду.

Процентиль (percentiles) - это способ группировки результатов по их проценту от всего набора образцов. Если 50% время отклика составляет 100 мс, это означает, что 50% запросов были возвращены в 100 мс или меньше.

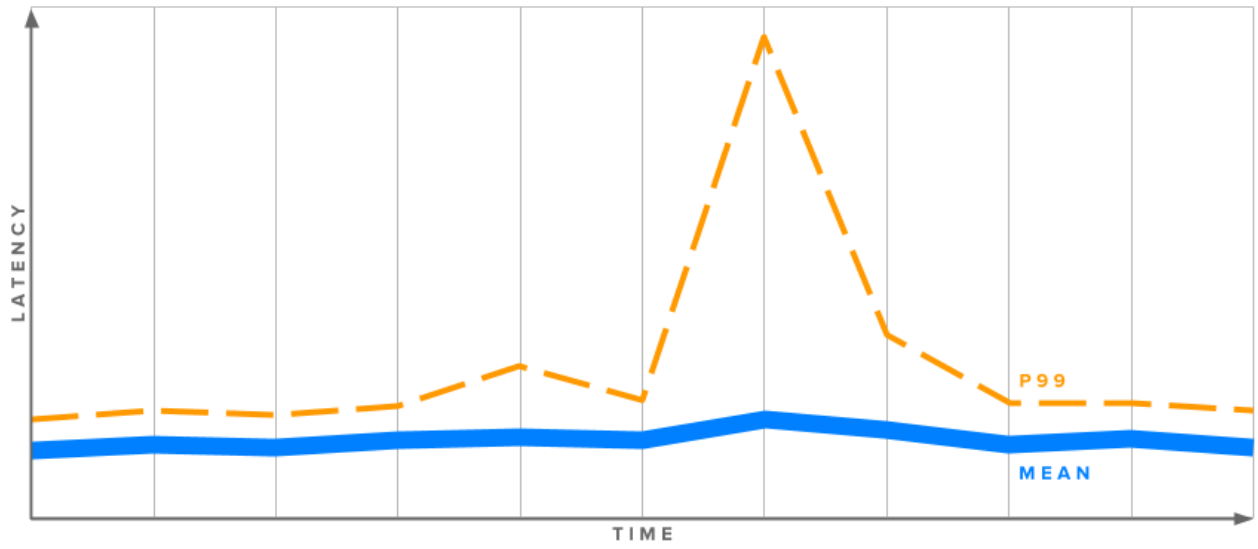

Рисунок 1 - Пример графика зависимости задержки от времени, показывающий большой всплеск в 99-м процентиле.

На приведенном выше графике (рисунок 1) показана задержка веб-сервера в течение определенного периода времени. Несмотря на то, что среднее (mean) время отклика довольно непротиворечивое, в 99-й процентильной линии имеется большой всплеск. Это означает, что 1% запросов пользователей выполнялся даже хуже, чем это 99-е

процентильное измерение, в то время как среднее значение оставалось относительно стабильным. По этой причине стоит взглянуть на процентили, чтобы получить более точное представление о том, что пользователи действительно испытывают.

Нагрузочное тестирование, на практике, это отправка имитированного HTTP-трафика на сервер для измерения производительности и ответа на некоторые важные вопросы, такие как:

- Имеет ли сервер достаточное количество ресурсов (ЦПУ, память и т.д.) для обработки ожидаемой нагрузки?
- Отвечает ли сервер достаточно быстро, чтобы обеспечить хороший пользовательский интерфейс?
- Эффективно ли работает приложение?
- Нужно ли масштабировать серверное оборудование?
- Существуют ли какие-либо особенно ресурсоемкие страницы или вызовы API?

Нагрузочное тестирование выполняется путем запуска программного обеспечения на одном компьютере (или в кластере машин) для генерации большого количества запросов на веб-сервер на втором компьютере (или другой более сложной инфраструктуре).

Программное обеспечение для нагрузочного тестирования, в основном, используется для поиска максимальных запросов в секунду, которые может обрабатывать сервер. Делается это путем отправки как можно большего количества запросов на сервер и просмотра того, сколько из них может успешно вернуться.

Это полезно в качестве первого шага для понимания максимальной мощности используемого сервера, но не дает много информации о времени отклика и фактической ежедневной производительности. Серьезно загруженный сервер может возвращать тысячу ответов в секунду, но, если каждый ответ будет занимает десять секунд, пользователи, вероятно, будут недовольны.

На приведенном ниже графике (рисунок 2) показана зависимость между пропускной способностью (количество ответов в секунду) и временем отклика.

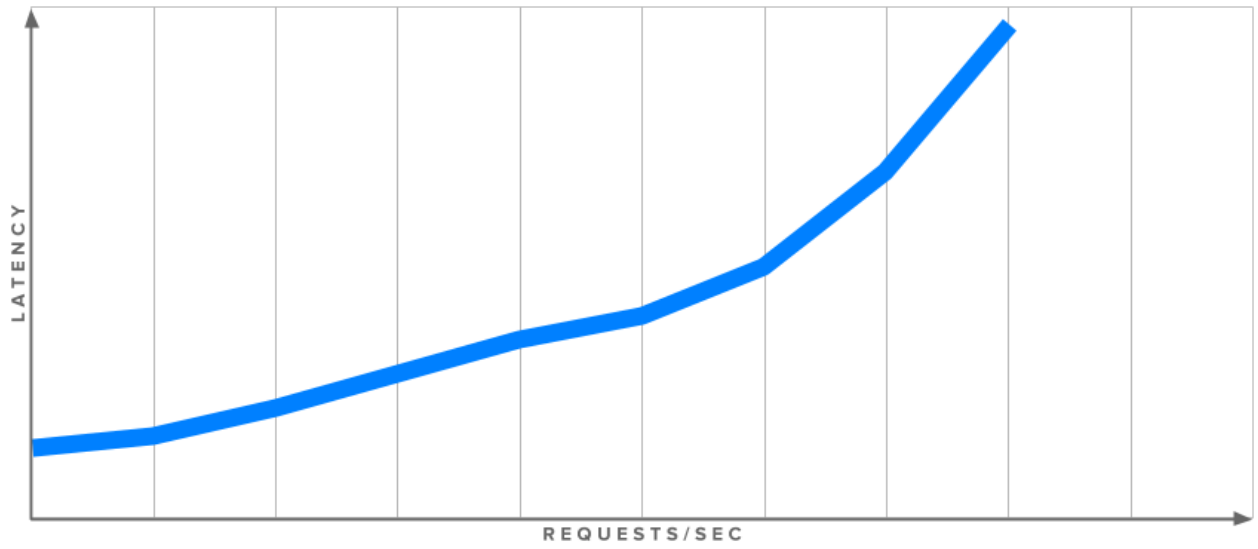

Рисунок 2 - Примерный график времени отклика и пропускной способностью, показывающий положительную корреляцию между ними

Это пример, и каждая конфигурация будет иметь уникальный профиль ответа, но общая тенденция заключается в том, что более высокая загрузка (больше запросов в секунду) приводит к более высокой задержке. Чтобы получить реальную информацию о задержке сервера при заданной нагрузке, нужно будет тестировать несколько раз с разными запросами.

Время загрузки веб-сайта в диапазоне 2-5 секунд является обычным явлением, но часть времени, которая связанна с задержкой веб-сервера, обычно составляет около 50-200 мс. Время отклика зависит от слишком большого количества факторов (основная аудитория, рынок, цель сайта, пользовательский интерфейс или служба API и т.д.). Невозможно дать более конкретное число, но главное иметь в виду, что каждый маленький бит скорости подсчитывается, и даже «незаметные» улучшения приводят к лучшим результатам.

2.2 Использование Apache JMeter для нагрузочного тестирования

Apache JMeter один из наиболее популярных и часто используемых инструментов нагрузочного тестирования. Изначально JMeter был разработан для тестирования Web и FTP приложений. Он часто используется для функционального тестирования, тестирования серверов баз данных и т.п. Для JMeter не требуется современная инфраструктура для тестирования нагрузки. Он обеспечивает поддержку нескольких инжекторов нагрузки, управляемых одним контроллером.

JMeter может интегрироваться с Selenium и Bean Shell для проведения автоматического тестирования.

На данный момент, он может использоваться в 3 режимах: GUI Mode, Server Mode, и Command Line Mode. Далее JMeter будет использоваться в GUI Mode.

На рисунке снизу (рисунок 3) изображен рабочий процесс JMeter.

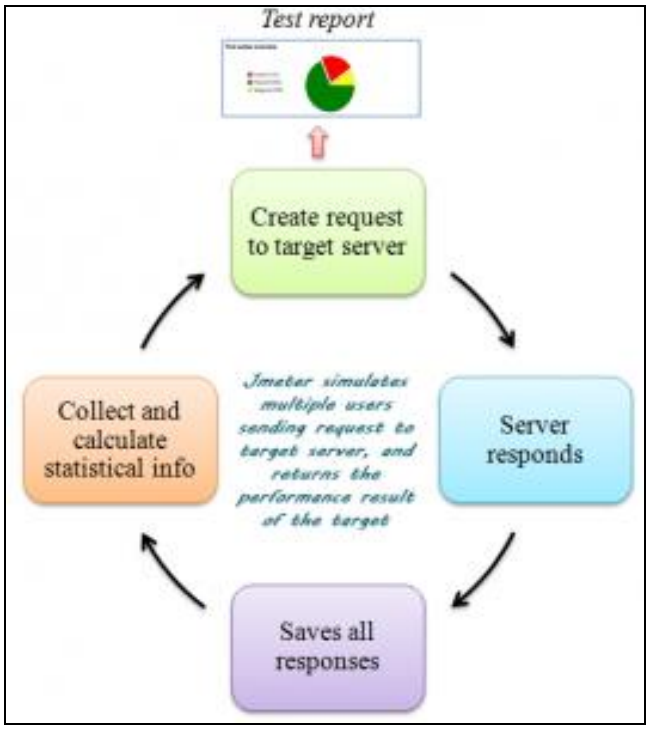

Рисунок 3 – Рабочий процесс JMeter

2.3 Установка JMeter

Список программного обеспечения, необходимого для запуска JMeter:

- Oracle Java or OpenJDK (6 or later)
- Apache JMeter

В данном руководстве были использованы следующие версии:

- Oracle Java 8 update 60, 64-bit
- JMeter 3.3

2.4 Построение базового плана тестирования

Исполняемый файл ApacheJMeter.jar находится в корневом каталоге, в директории bin.

После запуска JMeter можно увидеть графический интерфейс программы с пустым тестовым планом (рисунок 4).

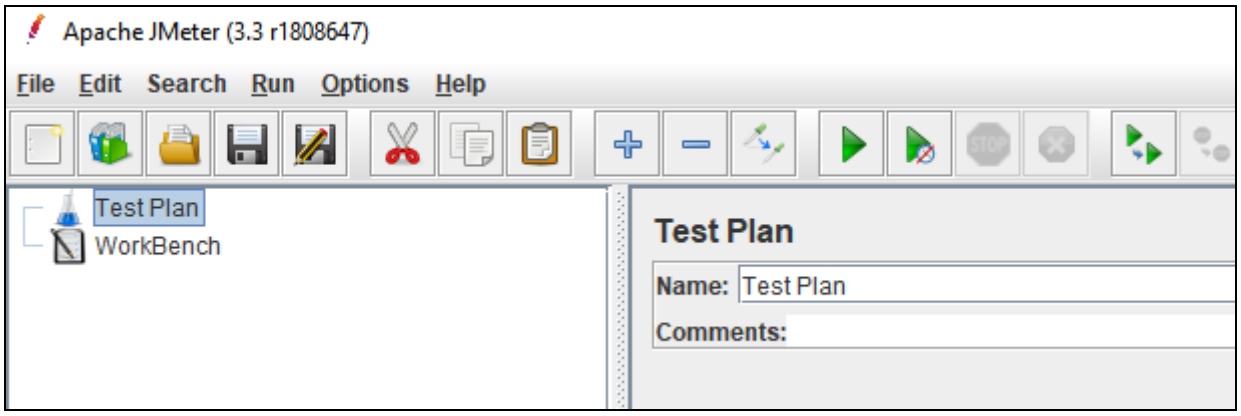

Рисунок 4 - Интерфейс JMeter

План тестирования состоит из последовательности тестовых сценариев, т.е. набор различных действий для создания нагрузки на сайт.

2.5 Создание Thread group

Результаты тестирования можно записать в поток (Thread). Добавление Thread в план тестирования показано на рисунке 5.

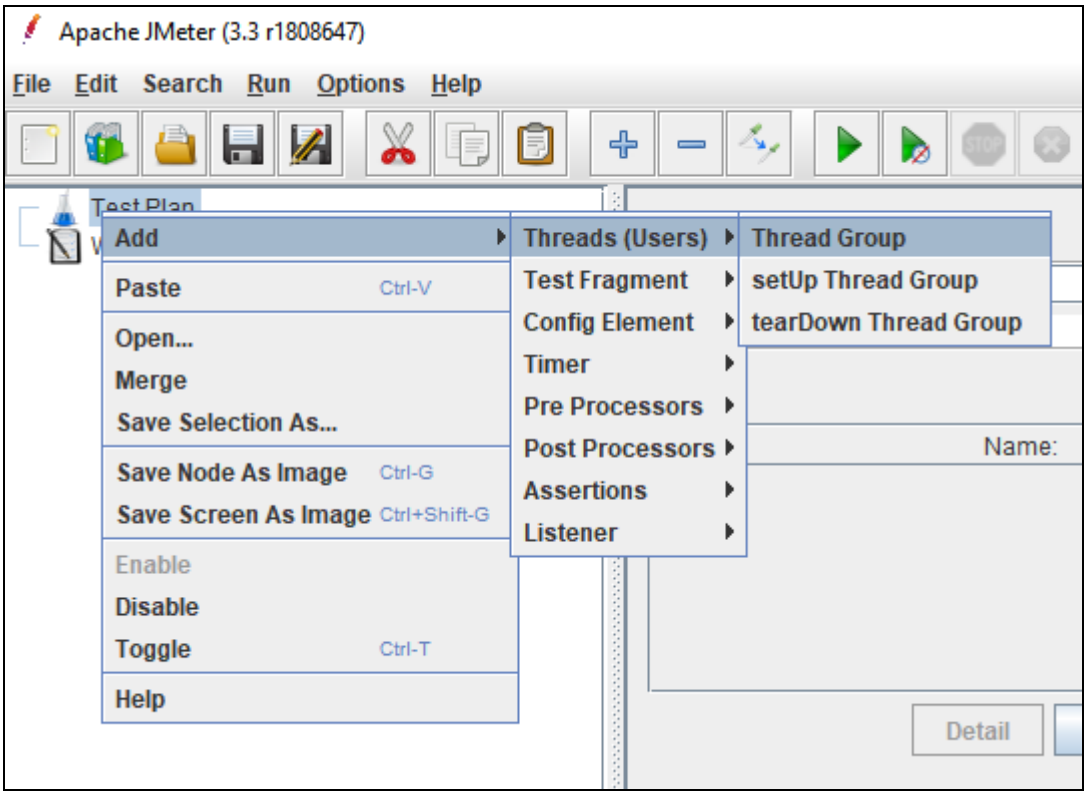

Рисунок 5 – Создание Thread group

Thread group (рисунок 6) имеет три особенно важных свойства, влияющих на нагрузку:

- Число потоков (пользователей): количество пользователей, которые JMeter попытается имитировать. Установите значение 50.
- Период разгона (в секундах): продолжительность времени, в течение которого JMeter будет загружать потоки для запуска. Установите значение 10.
- Количество циклов: количество раз, чтобы выполнить тест. Оставьте этот набор равным 1.

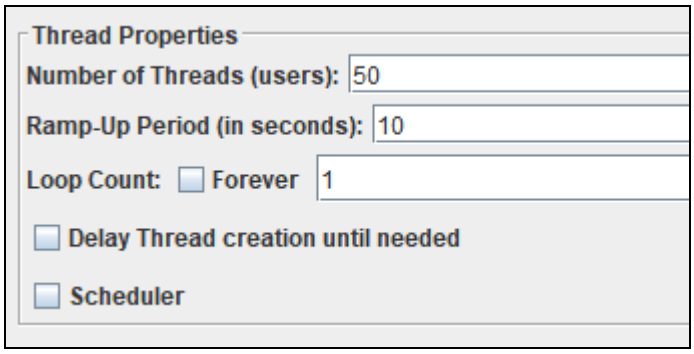

Рисунок 6 - Свойства Thread group

Исходя из введенных данных, тестирование будет проходить в 50 потоков, будет длиться 1 цикл. Время прогрева - 10 сек. Это значит, что потоки будут равномерно стартовать в течении 10 секунд, т.е. каждую секунду, будет запускаться новый поток.

2.6 Добавление HTTP Request Sampler

Теперь нужно добавить образец запроса HTTP в группу потоков (рисунок 7), который представляет страницу запроса, к которой будет обращаться каждый поток (пользователь).

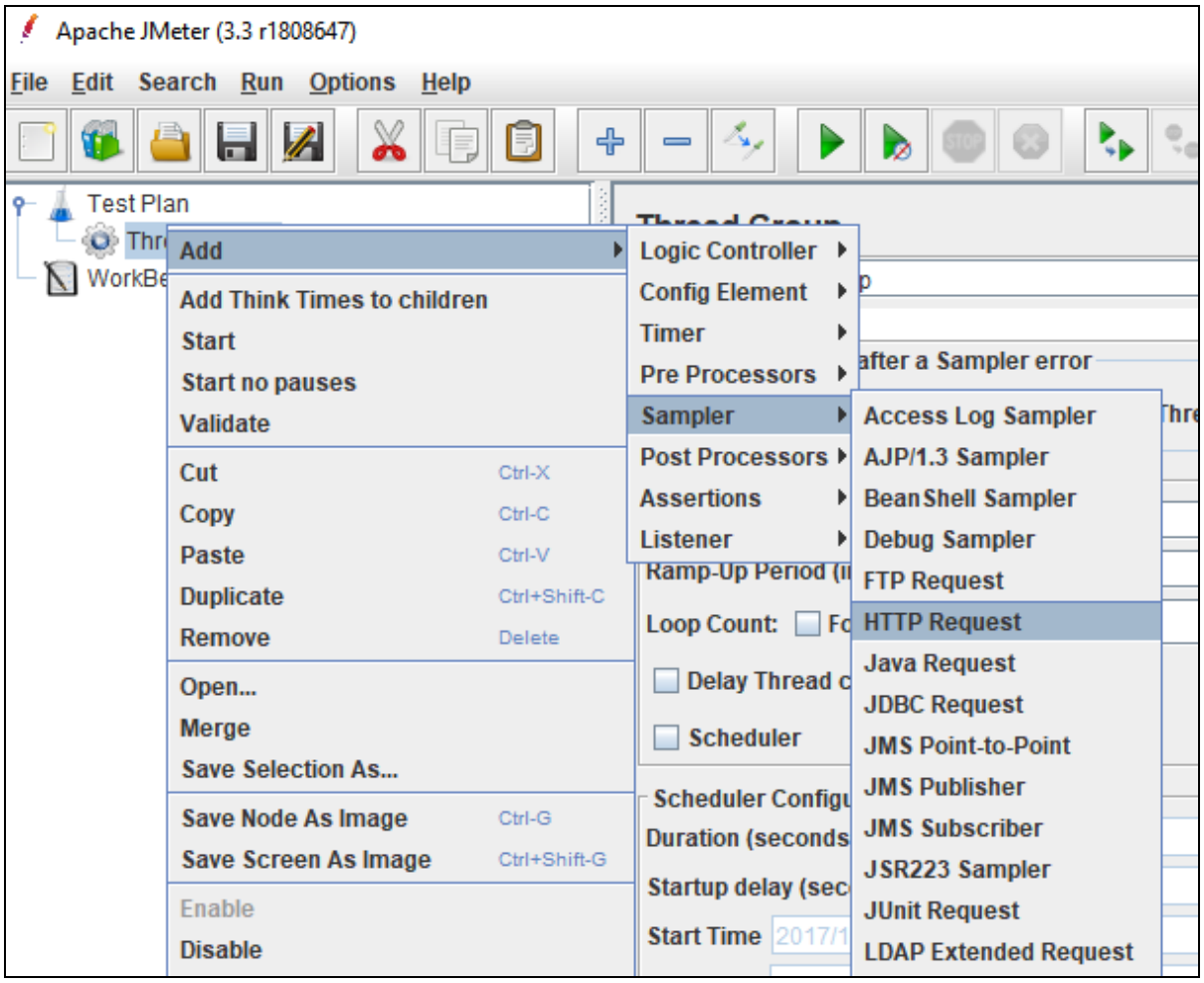

Рисунок 7 – Создание HTTP запроса

В HTTP-запросе в разделе «HTTP Request» нужно заполнить путь с элементом, который нужно запросить для каждого потока (пользователя). Установить его в https://do.swsu.org и каждый поток будет обращаться к этой странице.

2.7 Добавление результатов в таблицу

В JMeter слушатели (listeners) используются для вывода результатов нагрузочного тестирования. Существует множество слушателей, и другие

слушатели могут быть добавлены путем установки плагинов. Далее, будет использоваться таблица, потому что ее легко читать.

Thread Group → Add → Listener → View Results in Table

2.8 Запуск плана тестирования

Сначала нужно сохранить план тестирования, нажав «Файл», затем «Сохранить», затем указать нужное имя файла.

«Просмотреть результаты в таблице» → «Выполнить» → «Пуск» (или просто нажать зеленую стрелку «Start» в главном меню).

Результаты теста будут выведены в таблицу (рисунок 8).

Sample #	Start Time	Thread Name	Label	Sample Time(...	Status	Bytes	Sent Bytes	Latency	Connect Time(...
1	17:35:40.663	Thread Group ...	HTTP Request	2145		47212	107	1519	221
2	17:35:39.371	Thread Group ...	HTTP Request	3658		47212	107	3644	1481
3	17:35:39.591	Thread Group ...	HTTP Request	3448		47212	107	2957	1261
4	17:35:40.379	Thread Group ...	HTTP Request	2668		47212	107	2223	473
5	17:35:39.389	Thread Group ...	HTTP Request	3670		47212	107	3226	1585
6	17:35:39.979	Thread Group ...	HTTP Request	3128		47212	107	2748	873
7	17:35:41.035	Thread Group ...	HTTP Request	2124		47212	107	1967	227
8	17:35:39.777	Thread Group ...	HTTP Request	3433		47212	107	2678	1075
9	17:35:40.802	Thread Group ...	HTTP Request	2471		47212	107	2001	247
10	17:35:40.180	Thread Group ...	HTTP Request	3100		47212	107	2242	672
11	17:35:42.218	Thread Group ...	HTTP Request	2348		47213	107	2139	1136

Рисунок 8 - Результаты тестирования в табличном виде

Статус всех запросов - «Успех» (обозначается зеленым треугольником с галочкой в нем). Столбцы, которые, наиболее интересны будут Sample Time (ms) и Latency (не отображаются в примере).

- Время отклика (latency**)**: число миллисекунд, прошедшее между тем, когда JMeter отправил запрос и когда был получен первоначальный ответ

- Время выборки (sample time): количество миллисекунд, которое сервер принял для полного обслуживания запроса (время запроса + время отклика)

2.9 Автоматическая запись скрипта

В некоторых случаях, при тестировании веб-сервиса используются большие и нетривиальные скрипты для тестирования. Для того чтобы автоматизировать процесс записи и запуска подобных скриптов в JMeter имеется возможность записи скриптов.

Эмулируя работу прокси сервера, он будет записывать все полученные\отправляемые запросы.

2.10 Добавление Recording Controller

Для добавления Recording Controller, нужно выполнить следующие шаги (рисунок 9).

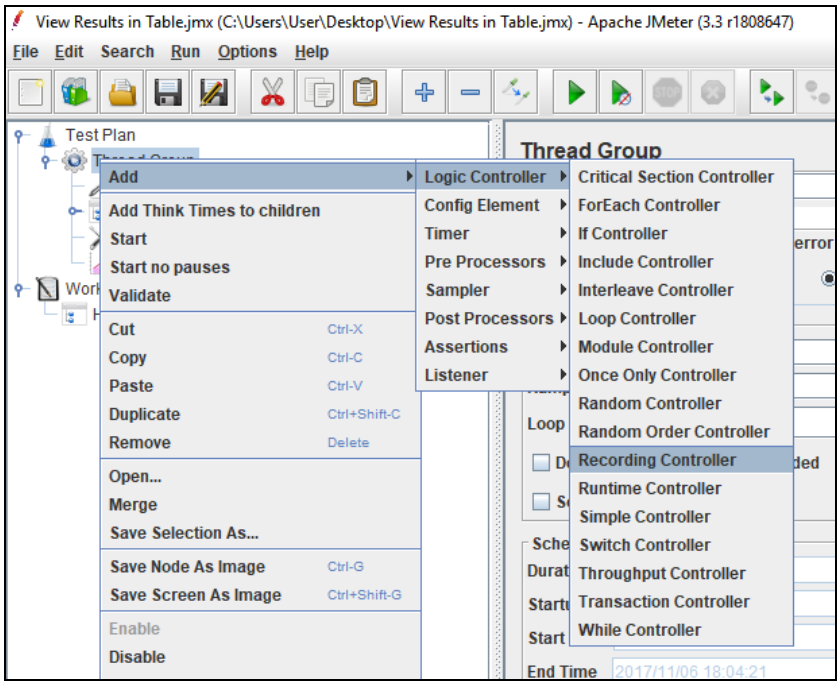

Рисунок 9 – Добавление Recording Controller

В данный элемент будут сохраняться все действия, которые будут происходить в браузере.

2.11 Добавление HTTP Cookie Manager

Добавление HTTP Cookie Manager происходит следующим образом:

Thread Group -> Add -> Config Element -> HTTP Cookie Manager

С помощью этого элемента будет реализована работа с сессиями через cookie. В настройках данного элемента нужно установить параметр Cookie Policy равным compatibility (рисунок 9).

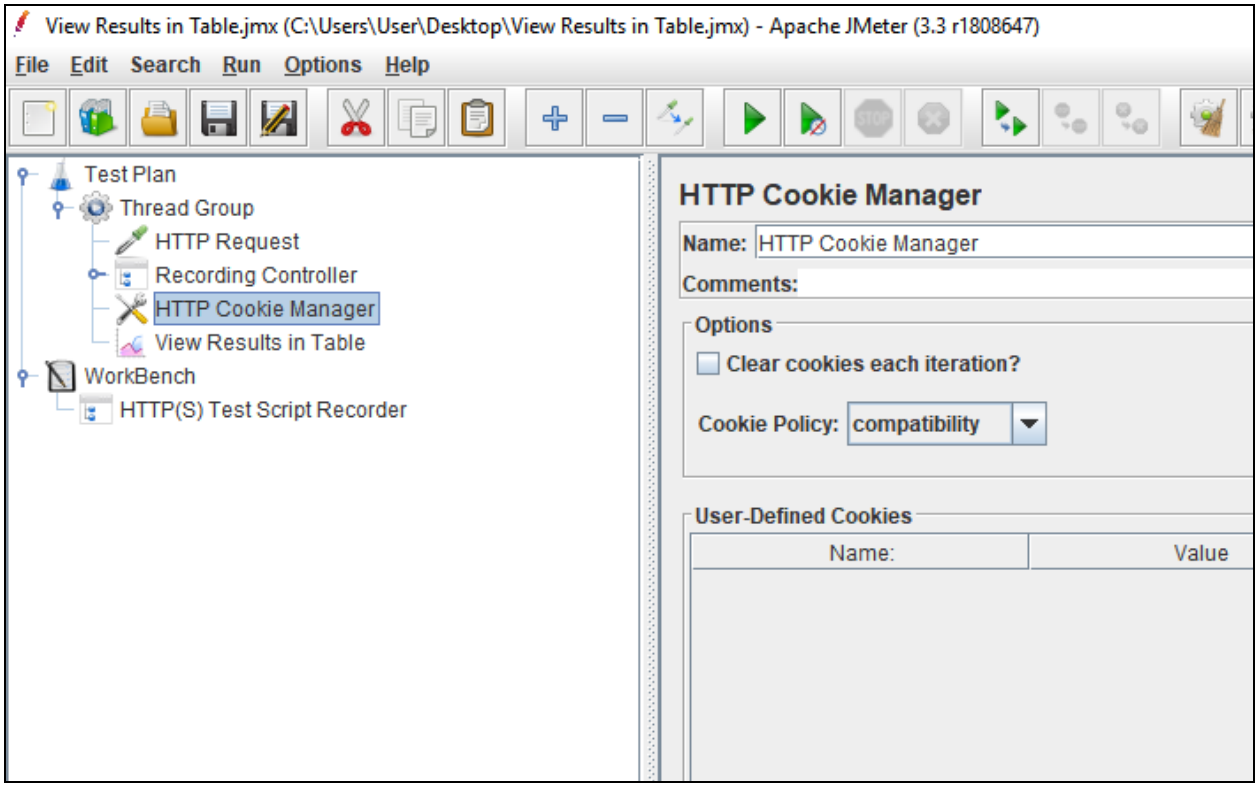

Рисунок 9 – Добавление HTTP Cookie Manager

2.12 Добавление HTTP Script Recorder

Добавление HTTP Cookie Manager происходит следующим образом (рисунок 10).

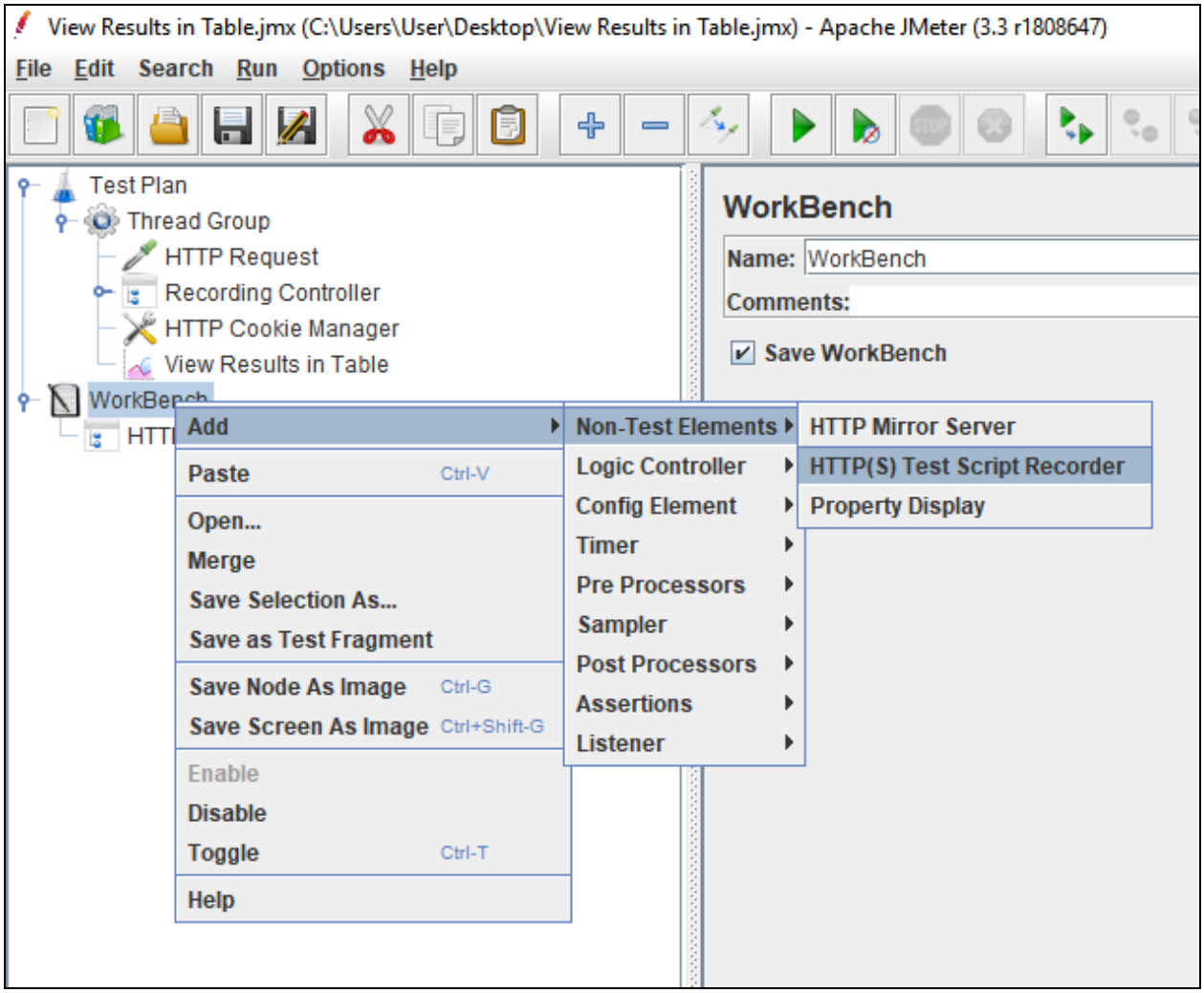

Рисунок 10 – Добавление HTTP Script recorder

Добавлять нужно в раздел WorkBench, так как непосредственно в ходе тестирования этот элемент не будет принимать участия. Он нужен только для того, чтобы создать сценарии тестов. Здесь имеется множество настроек данного элемента (рисунок 11).

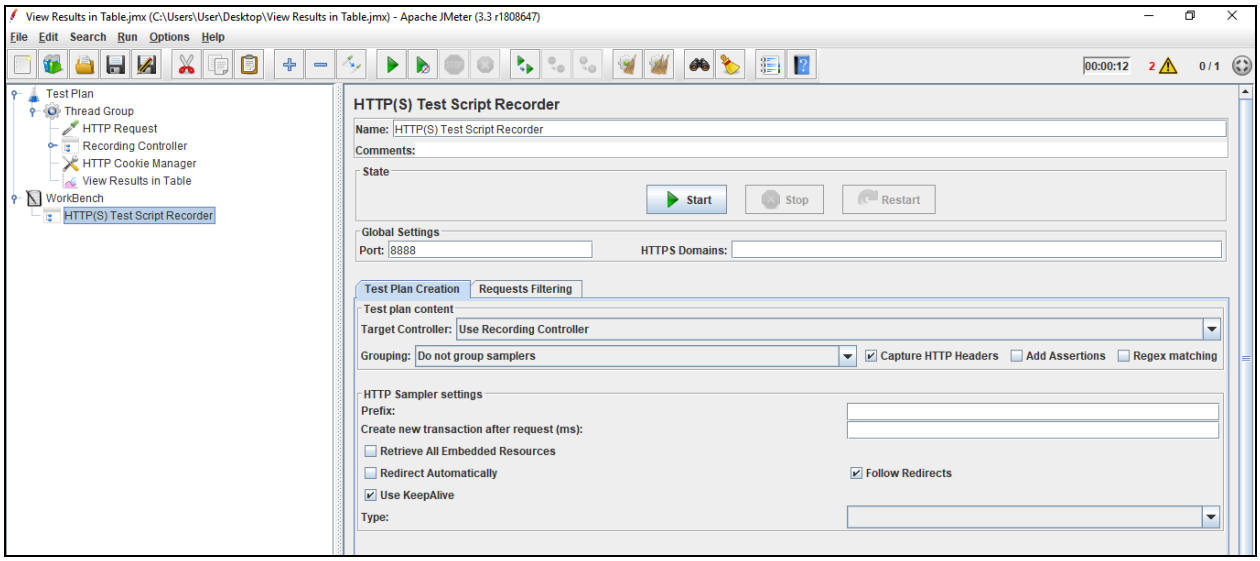

Рисунок 11 – Свойства HTTP Script Recorder

Здесь нужно изменить номер порта прокси-сервера, если порт по умолчанию 8888 занят, например, можно поставить 8080 или 8089.

Далее, в настройках браузера нужно указать адрес прокси и порт.

В качестве примера был использован браузер Google Chrome.

В настройках браузера нужно найти настройки прокси (рисунок 12).

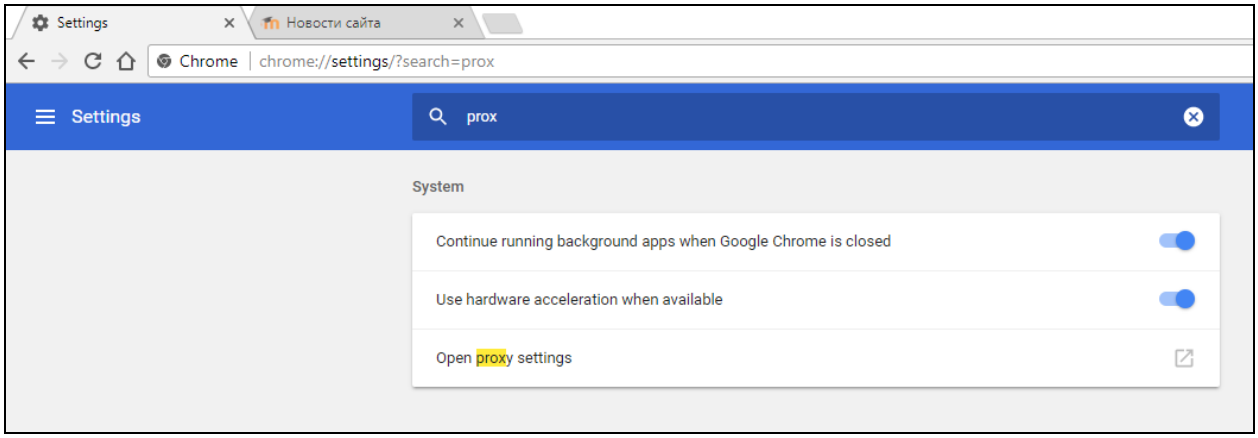

Рисунок 12 – Настройки прокси в браузере Google Chrome

В настройках прокси устанавливается локальный IP-адрес и порт (рисунок 13).

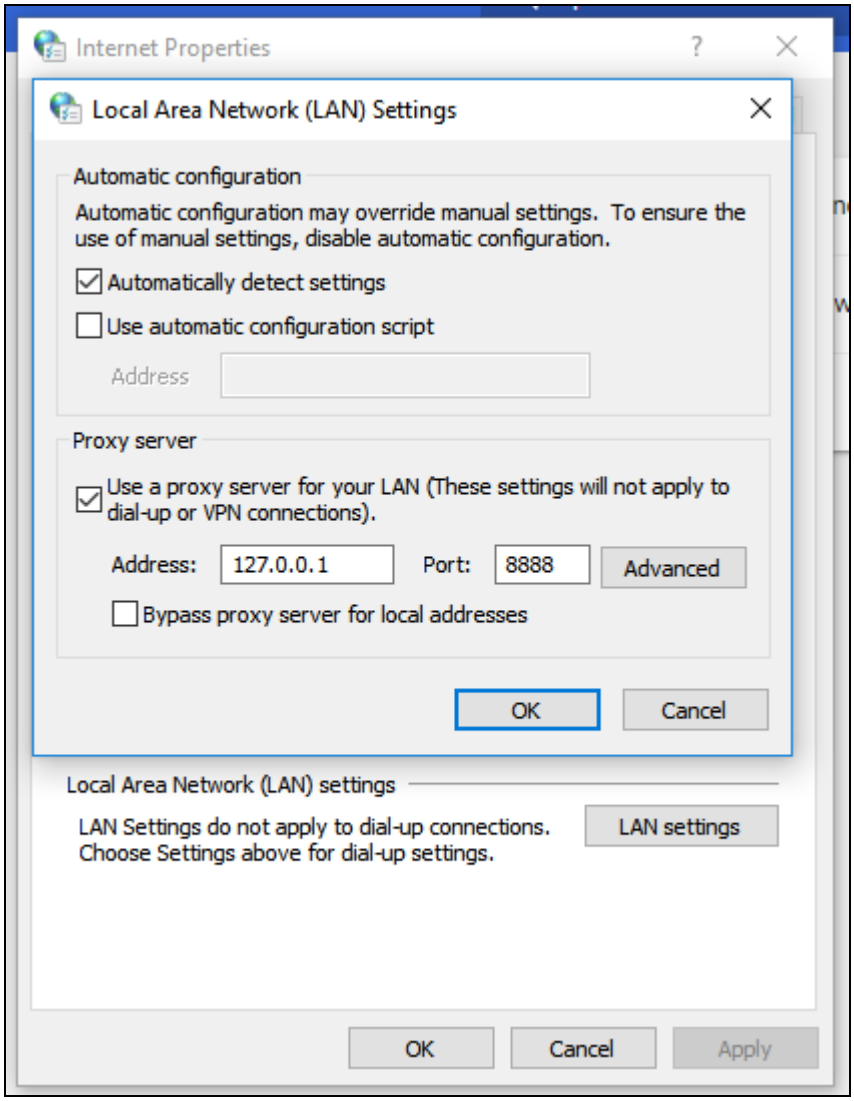

Рисунок 13 – Установка IP-адреса и порта для прокси

Убедиться, что браузер обращается к сети Интернет именно через него, для этого перейти на тестируемый сайт, при этом страница не должна загрузиться.

Далее, в JMeter нужно нажать кнопку Start в HTTP Script Recorder.

После этого, через браузер, открываем страницу тестируемого сайта, выполняем различные действия, посещаем разных страницы, при этом сценарий уже начнет сохраняться в элемент Recording Controller. По завершению выполнения нужных действий остановите Script Recorder.

Тестируемые скрипты располагаются в разделе Recording Controller (разработка 14).

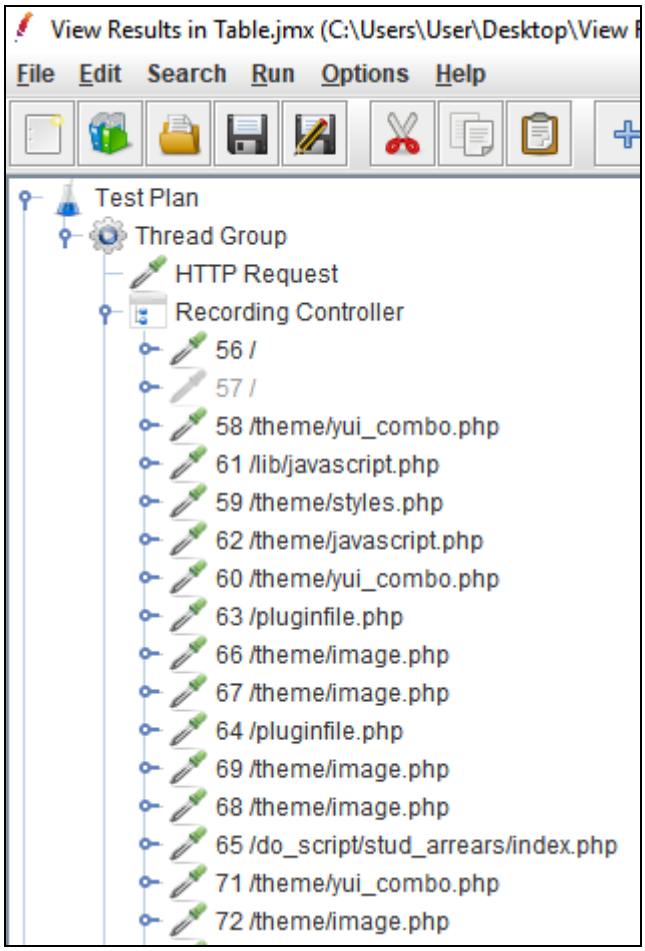

Рисунок 14 – Тестируемые скрипты

2.13 Добавление View Result in Table

Чтобы наблюдать результаты тестов, а также следить за ходом выполнения, нужно добавить элемент мониторинга. Для примера взят View Results in Table (рисунок 15).

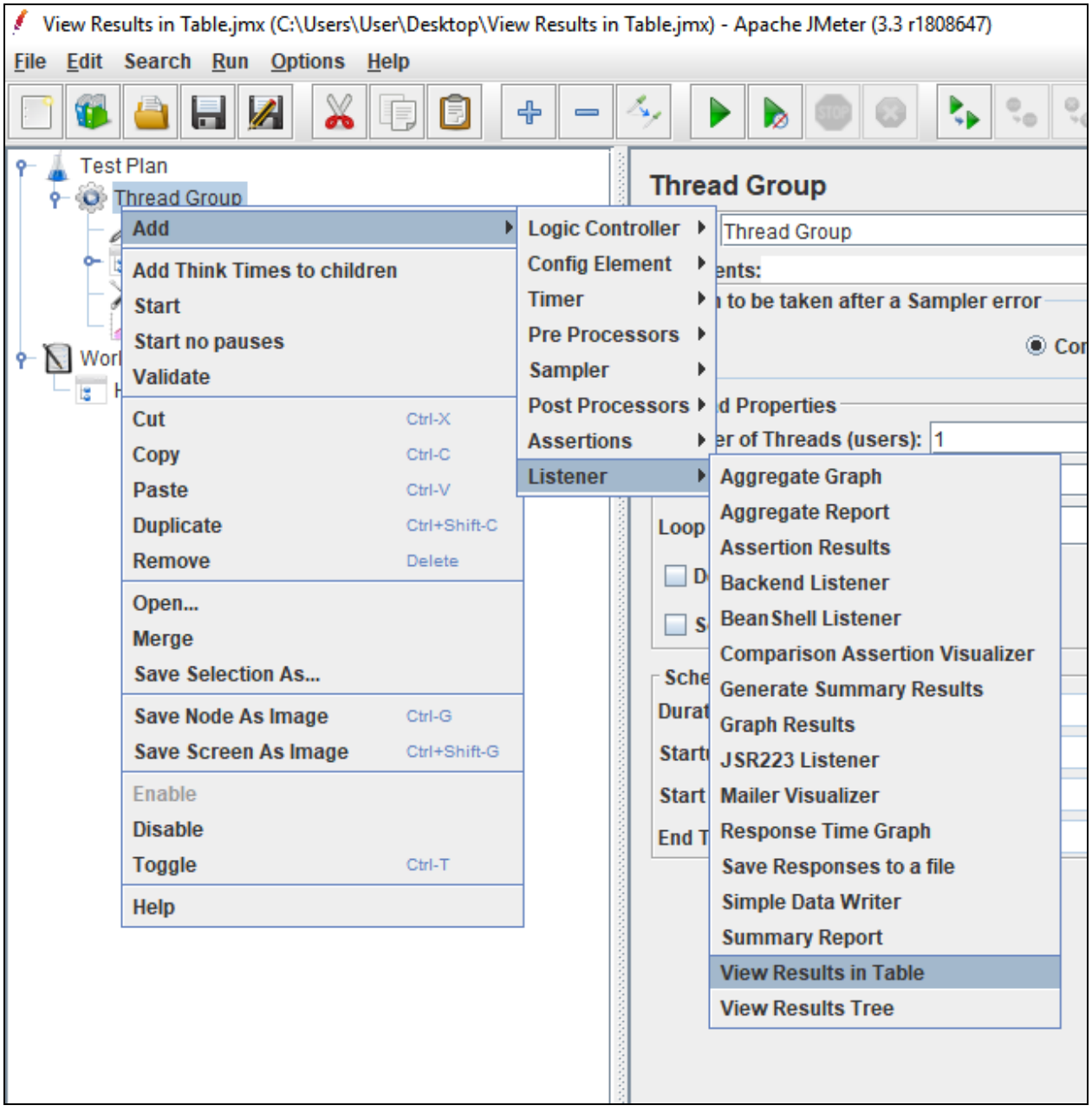

Рисунок 15 – Добавление View Result in Table

2.14 Запуск автоматического тестирования

Запуск автоматического тестирования осуществляется нажатием на кнопку Start в панели управления JMeter (рисунок 16).

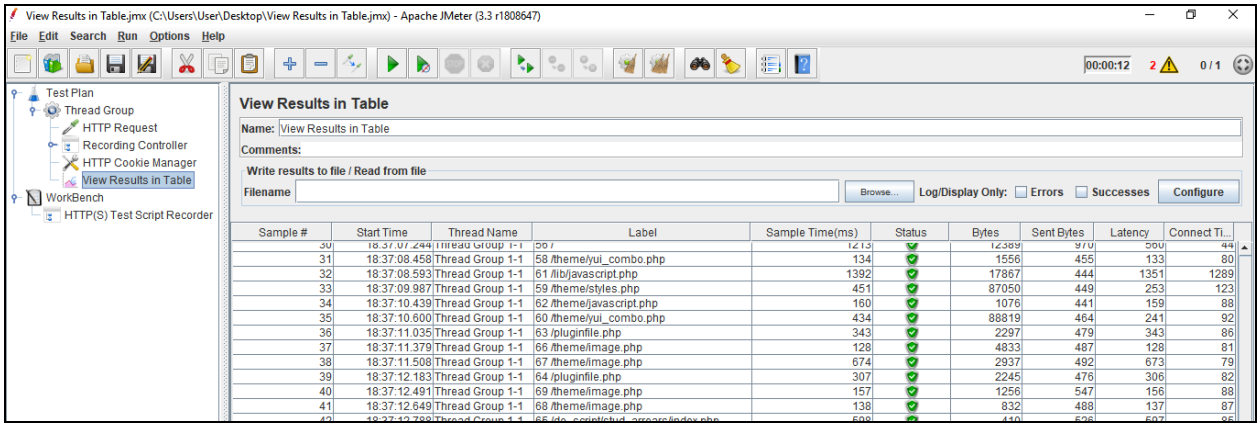

Рисунок 16 – Результаты тестирования

3 Содержание отчета по лабораторной работе

Отчет по лабораторным работам содержит отображение порядка выполнения и результатов лабораторной работы в JMeter для индивидуального задания.

4 Вопросы к защите лабораторной работы

1. Для чего нужна программа JMeter?
2. Что такое «число потоков»?
3. Каким образом осуществляется автоматическая запись скриптов для нагрузочного тестирования веб-приложения?
4. Для каких ключей создаются индексы?
5. Почему HTTP Script Recorder добавляется в раздел WorkBench?