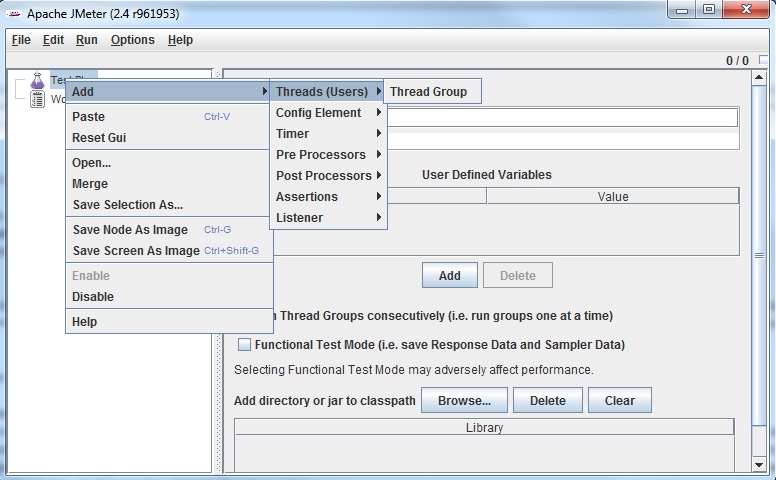
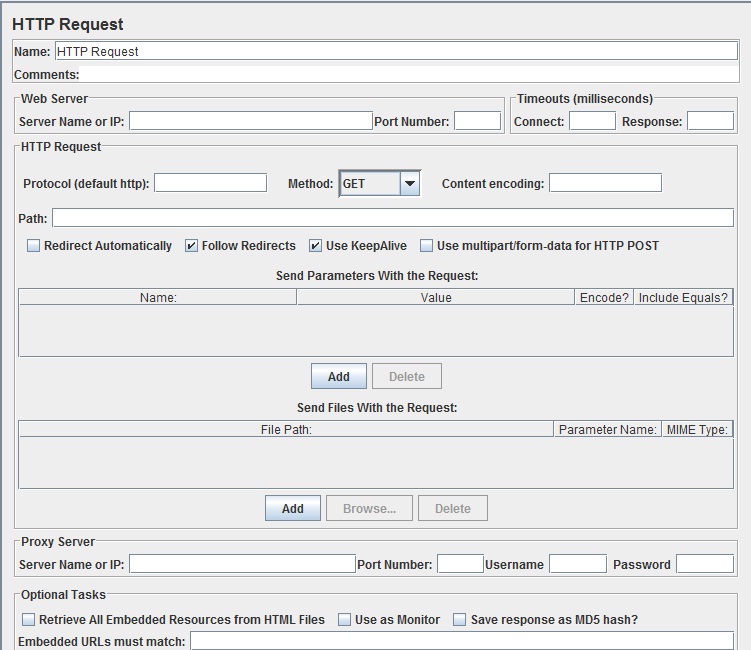
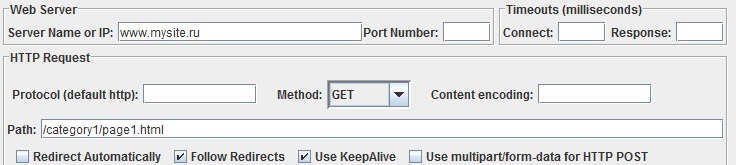
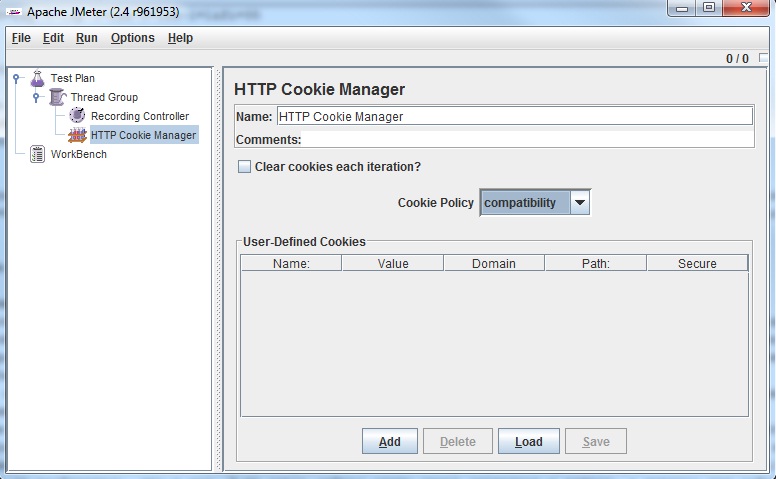
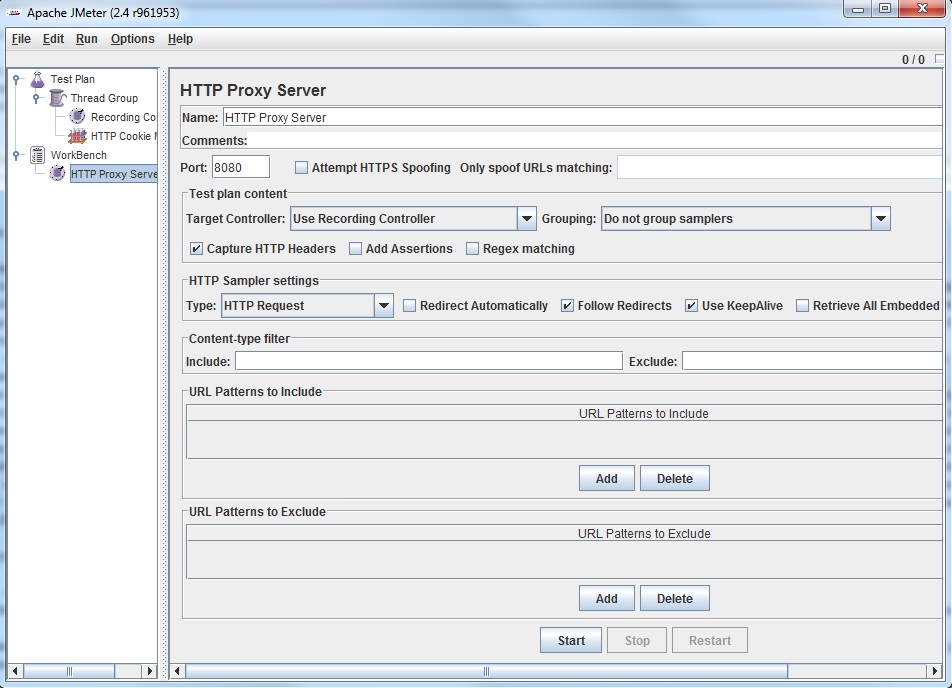
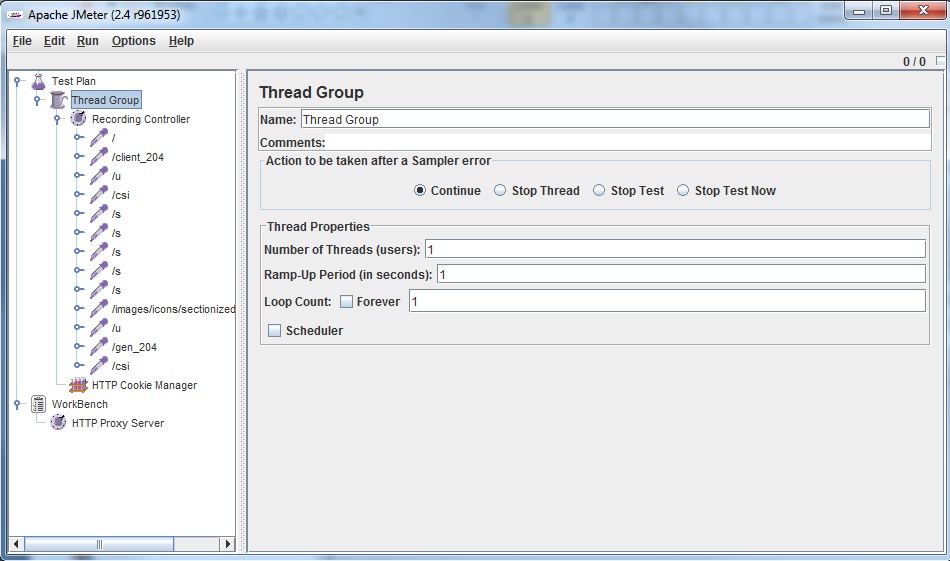
Для любого программного приложения, предназначенного для массового обслуживания пользователей, необходимо проводить нагрузочное тестирование на предмет его надежности и отказоустойчивости. А так как любой web-сайт — это по своей сути система массового обслуживания, то проверка его на отказоустойчивость всегда является неотъемлемой частью разработки. Существуют различные решения для проведения нагрузочного тестирования веб-приложений. Я не буду сейчас описывать их подробно, про некоторые из них есть упоминания [здесь](http://habrahabr.ru/blogs/hi/77593/).  
  
  
  
В этой статье я хочу поделиться своим опытом использования такого средства, как Apache JMeter. После того как мною были перепробовано с десяток различных подобных инструментом, в итоге я остановился именно на JMeter, так как его возможности с лихвой охватывали мои цели и задачи. И при этом данное программное средство весьма быстрое и легковесное.  
  
  
  
Для тех кто ни разу не использовал JMeter, рекомендую для начала почитать базовые обзоры, например, [Простой нагрузочный тест с Apache JMeter](http://habrahabr.ru/blogs/testing/84190/). Когда я первый раз запустил данную программу, первая мысль была разобраться во всем методом «тыка», но как выяснилось это вообще нереально, и метод «тыка» неприменим к JMeter. Поэтому если хотите его использовать, то сразу открывайте [мануал](http://jakarta.apache.org/jmeter/usermanual/index.html), поверьте, вам придется заглядывать туда очень часто, пока полностью не разберетесь, что и как. Я же здесь сейчас опишу самое очевидное и важное, а именно: как собственно создавать нагрузочные тесты. Если бы я в свое время сразу нашел подобную статью, то сэкономил бы без малого день на изучении этой софтины.   
  
  
  
Итак, поехали! Для интереса представим, что у нас не просто сайт, а серьезное веб-приложение с авторизацией, различными пост-запросами и т.д.  
  
  
  
1. Запускаем JMeter.  
  
  
  
2. Создаем Thread group:  
  
  
  
  
  
  
  
Это у нас как основной workflow, в котором мы будет записывать сценарии, добавлять различную логику и элементы управления.  
  
Теперь нужно создать собственно сценарий теста, т.е. набор различных действий для создания нагрузки на сайт. Можно создавать сценарий вручную. Это очень просто, 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. Например, так:  
  
  
  
  
  
  
  
Path

Кроме того можно добавить параметры для запросов. И такой способ создания сценариев теста вполне приемлем для несложных сайтов. Однако в нашем случае прописывать параметры запросов для авторизации может быть утомительным занятием. Кроме того, есть нужно создать последовательность действий для обхода сайта, скажем, хотя бы по 15-ти страницам, то опять же такой вариант уже не подходит. Конечно же ручной труд не для нас, и по-хорошему это дело нужно автоматизировать. В JMeter есть такая возможность, 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.  
  
  
  
3. Добавляем Recording Controller (Thread Group -> Add -> Logic Controller -> Recording Controller). В данный элемент будут сохраняться все наши действия, которые мы будет делать в браузере.  
  
  
  
4. Сразу добавим HTTP Cookie Manager (Thread Group -> Add -> Config Element -> HTTP Cookie Manager). С помощью этого элемента будет реализована работа с сессиями через cookie. В настройках данного элемента устанавливаем параметр Cookie Policy = compatibility. За описанием в мануал. Получили такую картину  
  
  
  
  
  
  
  
5. Добавляем элемент HTTP Proxy Server. Добавлять его надо в раздел WorkBench (WorkBench -> Add -> Non-Test Elements -> HTTP Proxy Server), так как непосредственно в ходе тестирования этот элемент не будет принимать участия. Он нам нужен только, чтобы создать сценарии тестов. Здесь мы видим множество настроек данного элемента:

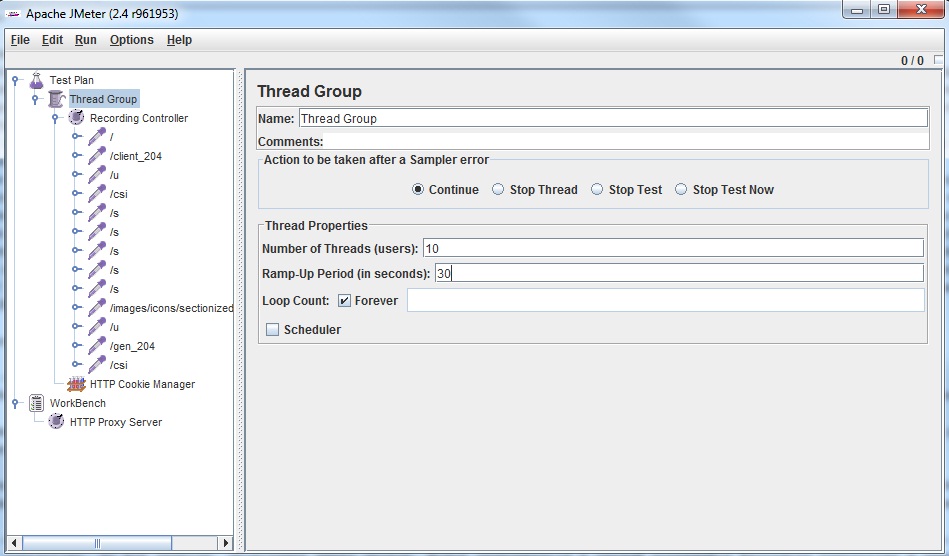
http://dml.compkaluga.ru/forum/style_images/ip.bitrix/img-resized.png Уменьшено: 84% от [ 952 на 688 ] — нажмите для просмотра полного изображения

  
  
  
  
По сути здесь достаточно только изменить номер порта прокси-сервера, если порт по умолчанию 8080 у вас уже занят, например, можно поставить 8089. Если оставить в графе Target Controller значение Use Recording Controller, то все запросы, проходящие через прокси, будут записываться в первый попавшийся Recording Controller в нашем тест-плане. Но так как на данный момент он там всего один, то нас этот вариант устроит. Далее рекомендую обратить внимание на настройку фильтрации. Эти настройки предоставляют широкие возможности по фильтрации запросов. Это могут быть отдельные страницы, или например все js-скрипты и т.д. и т.п. Опять же в мануале они описаны достаточно хорошо.  
  
  
  
6. В настройках браузера нужно указать адрес прокси и порт. Убедиться, что браузер ходит в интернет именно через нее, для этого перейти на любой сайт в интернете, при этом страница не должна загрузиться. Далее нам осталось в окне с настройками HTTP Proxy Server нажать на кнопку Start, запустив тем самым нашу проксю. После этого через браузер, открываем страницу тестируемого сайта, логинимся, выполняем различные действия, посещаем разных страницы, при этом сценарий уже начнет сохраняться в элемент Recording Controller. Когда вы поймете, что записанный вами сценарий уже достаточно суровый, для того чтобы задать вашему сайту жару, останавливаете прокси.  
  
Получили, примерно, следующую картину:

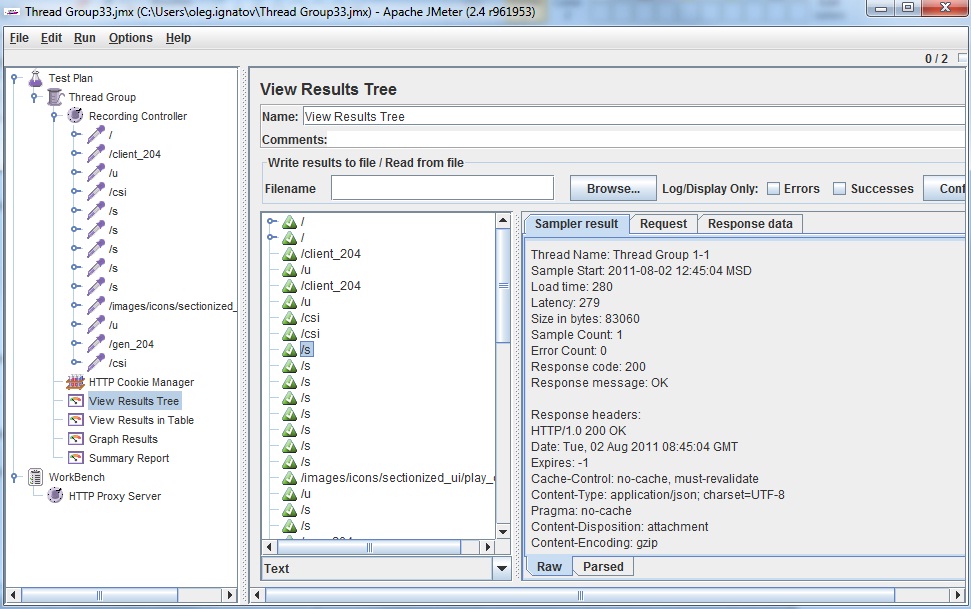
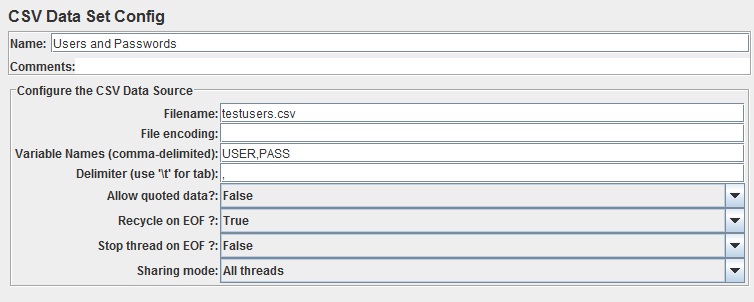
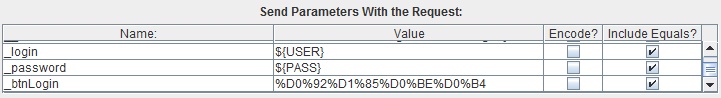
http://dml.compkaluga.ru/forum/style_images/ip.bitrix/img-resized.png Уменьшено: 84% от [ 950 на 561 ] — нажмите для просмотра полного изображения

  
  
  
  
Названия элементов могут оказаться мало о чем говорящими. Поэтому чтобы в дальнейшем было просто наблюдать за ходом теста, эти элементы имеет смысл переименовать.  
  
  
  
7. В настройках элемента Thread Group выберите количество потоков, число итераций, время прогрева. Например, я установил такие настройки:

http://dml.compkaluga.ru/forum/style_images/ip.bitrix/img-resized.png Уменьшено: 84% от [ 949 на 556 ] — нажмите для просмотра полного изображения

  
  
  
  
Это значит, что тестирование будет проходить в 10 потоков, будет длиться бесконечно пока, его принудительно не остановят. Время прогрева я поставил 30 сек. Это значит, что потоки будут равномерно стартовать в течении 30 секунд, т.е. каждые три секунды, будет запускаться новый поток.  
  
  
  
8. Чтобы наблюдать результаты тестов, а также следить за ходом выполнения, нужно добавить несколько элементов мониторинга. Я обычно добавляю такие: View Results Tree, View Results in Table, Graph Results, Summary Report. Как и что они показывают, думаю, вы разберетесь сами. Единственно, что отмечу, очень полезный элемент View Results Tree, в котором можно смотреть все параметры и содержимое запросов и ответов:

http://dml.compkaluga.ru/forum/style_images/ip.bitrix/img-resized.png Уменьшено: 82% от [ 971 на 609 ] — нажмите для просмотра полного изображения

  
  
  
  
Таким образом можно проверять, что всё работает так как надо.  
  
  
  
Вот собственно и всё! Таким образом можно создать нагрузочный тест для вашего сайта за считанные минуты.  
  
Напоследок, скажу еще пару полезных фич, которые реализованы в Apach JMeter.  
  
  
  
  
**Настройки по умолчанию**  
  
  
Если вы хотите задавать настройки для всех элементов HTTP Request, то это можно сделать, добавив специальный набор настроек HTTP Request Defaults (Recording Controller -> Add -> Config Element -> HTTP Request Defaults). С виду это окно очень похоже на HTTP Request. Все параметры, которые вы в нем зададите, будут использовать во всех остальных элементах данного набора. Это может быть удобно во многих ситуациях, например, когда у вас развернуто несколько вебов на разных портах, и вам периодически нужно тестировать то один, то другой. Чтобы не править один порт для всех запросов, меняем его всего лишь в одном месте в HTTP Request Defaults. Все остальные будут использовать его, если у них не задан свой.  
  
  
  
  
**Загрузка списка пользователей из файла**  
  
  
Данная задача также проста и очевидна. Если нужно натравить на ваш многострадальный веб-ресурс сразу целую свору разъяренных юзверей, которые будут хаотично нажимать на все кнопки подряд, то 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 (если положить его рядом с файлом проекта, то достаточно указать одно имя), а также имена переменных. Эти имена вы задаете сами, они в дальнейшем будут использоваться при формировании запроса. К примеру, для связки юзерпароль, я указал два имени переменных: USER, PASS.  
  
Сам файл с расширением csv может выглядеть так:  
  
*userName1,pass1  
  
userName2,pass2  
  
userName3,pass3*  
  
  
  
Отлично, юзеров мы подали на вход джиметру. Теперь нужно сказать ему, чтобы он их заюзал 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}.  
  
У меня это выглядит так:  
  
  
  
  
  
  
  
После этого JMeter будет циклически использовать эти креденшелы для разных потоков, обеспечивая таким образом вполне реалистичную нагрузку на сайт.  
  
  
  
На этом всё, всем спасибо за внимание!