Slash и backslash: вехи на пути

Немного истории

Slash

Возникновение слеша относят к временам Римской империи. На ранних стадиях современности, во Фрактуре [1], которая была широко распространена по всей Европе в средневековье, слеш (/) использовался вместо запятой, в то время как двойной слеш (//) использовался вместо тире. Двойной слеш, в конечном счете, превратился в символ похожий на знак равенства (=), а позже был еще больше упрощен до тире или дефиса [2].

Backslash

Боб Бемер ввел обратный слеш (\) в набор символов ASCII, 18 сентября 1961 года, как результат изучения частоты использования символов встречающихся в частности в программах на ALGOL'е. Тогда же вместе с обратным слешем в стандарт были включены и квадратные скобки.

В частности \ был введен, чтобы булевы операторы ALGOL'а AND и OR могли быть представлены с помощью ASCII символов как "/\" и "\/" соответственно [3,4].

Как же вышло, что исторически православный слеш заменился на свое зеркальное отображение, введенное как вспомогательный символ специально для уже мертвого языка?

Русскоязычная Википедия по этому говорит вот что:

В операционных системах DOS и Windows фирмы Microsoft и их аналогах других разработчиков, обратная косая используется для разделений имён директорий (каталогов) при указании пути к файлу. Прямая косая, применяемая для этого в Unix не могла быть использована в MS-DOS, потому что уже была задействована для указания ключей командной строки (оставшегося в наследство от CP/M, где MS-DOS команда «dir /w» писалась как «dir/w») [5].

Так как такое объяснение меня не слишком удовлетворило, пришлось найти статью «**Why is the DOS path character "\"?**» [6], которая вполне утолила моё любопытство. Вольный перевод избранных частей в моем исполнении:

То что символ "/" конфликтовал с разделителем пути другой относительно популярной ОС не был связан напрямую с разработчиками – в конце концов, DOS не поддерживал директорий, просто файлы в одном корневом каталоге.

*Для MS-DOS 2.0 (в котором появился поддержка каталогов), дизайнеры DOSa выбрали гибридную версию – у них уже были имена дисков в наследство от DOS 1.0, поэтому разработчикам пришлось их использовать. И в дополнение к именам дисков они решили использовать *nix-style метод определения иерархии каталогов — вместо использования каталога в имени файла (как это было сделано в VMS и DEC-20), они просто сделали каталог и имя файла неотъемлемыми частями пути. Но с этим была проблема. Невозможно было использовать разделитель пути *nix (/), по той причине что слэш уже использовался как разделитель ключей.*

Что им было делать? Они конечно могли использовать "." как в DEC, но точка уже использовалась как разделитель между именем файла и расширением. Поэтому они выбрали наилучший вариант из оставшихся — символ "\", который был визуально похож на "/".Таким вот образом и был выбран символ "\" для разделения путей в DOS.

*Кстати есть небольшой секрет про MS-DOS. Разработчики DOS не были довольны таким положением дел – они использовали Xenix [7] для почты и прочих вещей, поэтому они были знакомы со структурой *nix команд. Поэтому они добавили в ОС возможность принимать в качестве разделителя путей как "/" так и "\" (это работает и сегодня, кстати – попробуйте выполнить «notepad c:/boot.ini» под XP (если ваш пользователь имеет права админа)). Дальше — больше. Они добавили недокументированный системный вызов, чтобы изменить символ разделителя ключей. И обновили утилиты, чтобы те поддерживали этот флаг. Они даже добавили в config.sys параметр, SWITCHAR, который позволит пользователю установить разделитель ключей на "-". Таким образом можно было превратить MS-DOS в *nix-style ОС, используя "-switch", и пути с разделителем "/".*

Собственно к чему это все?

Меня побудила разобраться в этой теме следующая ситуация.

Была поставлена задача — наладить систему отчетов для автоматизированных тестов. Тесты у нас используются двух видов – Selenium (функциональные) и Jmeter (нагрузочные). Собственно в этом не было ничего сложного — для этих целей существует вполне себе open-source проект под названием logging selenium [8] и plugin для maven — chronos [9]. Настроив всё и протестировав отчеты локально, принялся за интеграцию с нашей CI — TeamCity. Вот тут-то меня и ждала та самая неожиданность, которая стала поводом для написания этой статьи.

После выполнения всех тестов отчет о Selenium-тестах имел следующий вид:

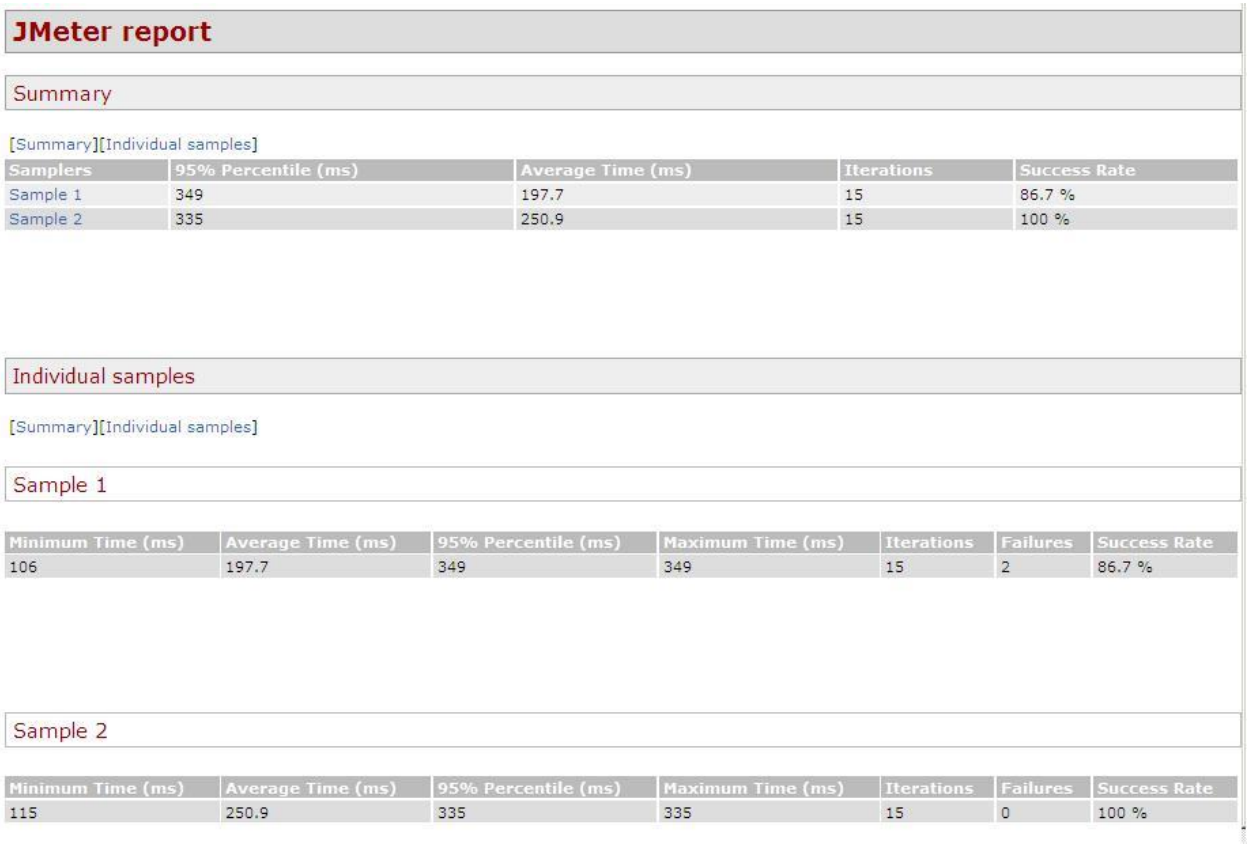
Test results

user-agent:	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.16) Gecko/20110319 Firefox/3.6.16
test-started:	10:53:25 25-05-2011
test-duration [ms]:	9593
commands processed:	14
verifications processed:	4
commands not logged:	{getHtmlSource}

Selenium-Command	Parameter-1	Parameter-2	Res.RC	Res.Selenium	Time [ms]	Calling-Class with Linenumber
executing setUp()						
windowMaximize			OK	OK	16	.tests.core.BaseSeleniumTest#61
windowFocus			OK	OK	0	.tests.core.BaseSeleniumTest#62
executing open()						
open	/		OK	OK	891	.tests.core.BaseSeleniumTest#118
executing typeText()						
type	q	night	OK	OK	31	.tests.core.BaseSeleniumTest#114
executing clickOn()						
click	btnG		OK	OK	63	.tests.core.BaseSeleniumTest#98
executing waitForElement()						
isElementPresent	btnG		OK	true	15	.tests.core.BaseSeleniumTest#86
executing takeScreenshot()						
					766	.tests.core.BaseSeleniumTest#133

Всё отлично отображалось, и никаких отличий от локальной версии не было.

Но вот отчет, который отобразился для Jmeter-тестов, воодушевления не вызывал:



JMeter report

Summary

[Summary][Individual samples]

Samplers	95% Percentile (ms)	Average Time (ms)	Iterations	Success Rate
Sample 1	349	197.7	15	86.7 %
Sample 2	335	250.9	15	100 %

Individual samples

[Summary][Individual samples]

Sample 1

Minimum Time (ms)	Average Time (ms)	95% Percentile (ms)	Maximum Time (ms)	Iterations	Failures	Success Rate
106	197.7	349	349	15	2	86.7 %

Sample 2

Minimum Time (ms)	Average Time (ms)	95% Percentile (ms)	Maximum Time (ms)	Iterations	Failures	Success Rate
115	250.9	335	335	15	0	100 %

Напрочь отсутствовали все изображения на странице.

После просмотра исходного кода страницы стало понятно, что во всем виноват backslash. Ссылки на изображения были указаны в таком формате:

```
<img src="images\gc-performancetest.png">
```

Справедливости ради стоит заметить, что изображения отсутствовали в Firefox, но прекрасно отображались в IE. Хотя если бы IE не отображал ресурсы в URI которых встречается обратный слеш, как разделитель пути для Windows, то в, и не без того подпорченной, репутации индийских программистов образовалась бы еще одна брешь.

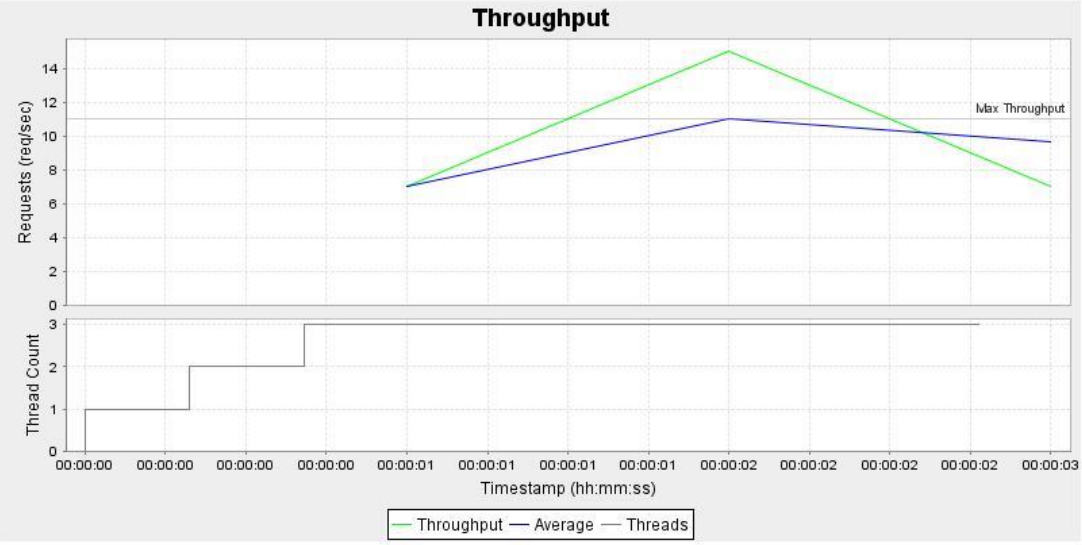
В общем, после недолгих раздумий стало понятно, что где-то внутри chronos'a используется File.separator, который и повинен в появлении обратного слеша в путях ресурсов и проблемах с отображением графиков в отчетах. Скачивание исходников плагина и исправление соответствующей строки, в которой формировался URL, решило проблему и дало возможность насладится прекрасными графиками, которые генерируются на основе результатов выполнения JMeter-тестов.

JMeter report

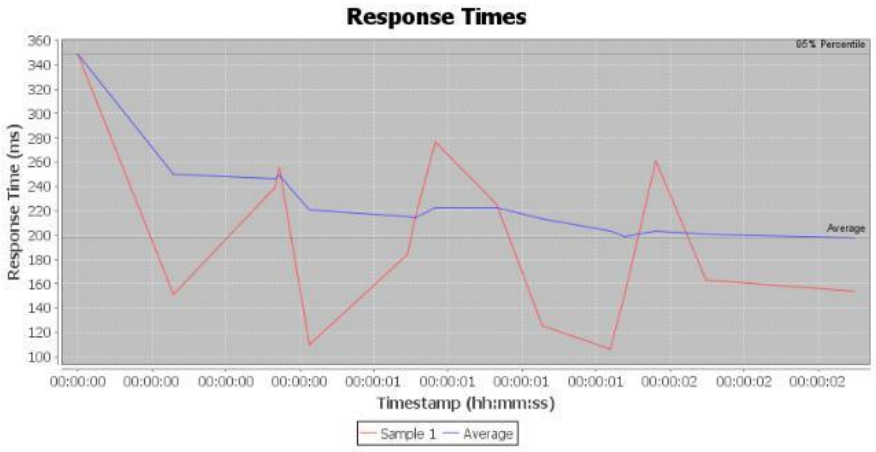
Summary

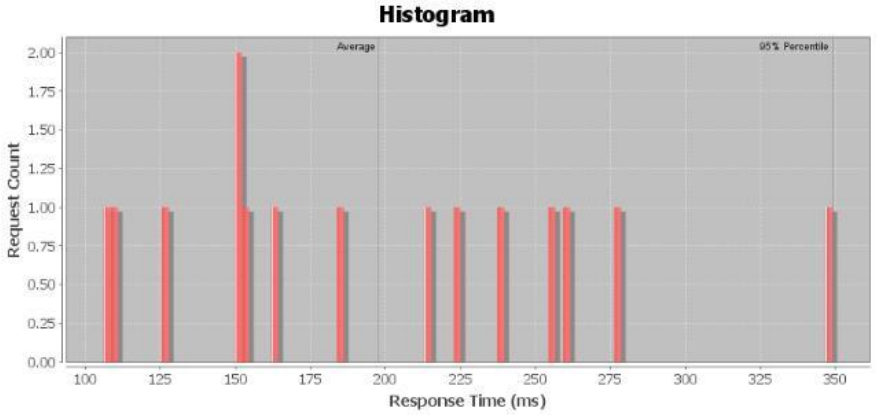
[Summary][Individual samples]

Samplers	95% Percentile (ms)	Average Time (ms)	Iterations	Success Rate
Sample 1	349	197.7	15	86.7 %
Sample 2	335	250.9	15	100 %



Minimum Time (ms)	Average Time (ms)	95% Percentile (ms)	Maximum Time (ms)	Iterations	Failures	Success Rate
106	197.7	349	349	15	2	86.7 %





Заключение

В заключении хотелось бы предостеречь от бездумного использования нативного File.separator – это не всегда приводит к кроссплатформенности, а в некоторых случаях даже может стать причиной появления новых багов. Обычный же слеш работает в Windows (зачастую), работает в *nix, Java и наконец его стоит уважать хотя бы по праву старшинства, так как он на полторы тысячи лет старше своего зазеркального брата.

Материал взят из источника – «Slash и backslash: вехи на пути / Хабр»: https://habr.com/ru/post/120652/