http://vqs.markelov.biz

Система управления сайтами

Wacko Wiki Very Quick Start

Описание системы

Разработчики: Илья Маркелов
Антон Кузнецов

Москва, Королёв 2009-2014

Оглавление

О системе

Кратко

Wacko Wiki Very Quick Start – это движок для быстрой разработки Web-приложений на базе PHP 5/MySQL.

Цели и задачи

Система VQS разрабатывается с 2009 года как универсальный инструмент для создания web-приложений. Изначально представляла собой адаптацию CMS WWQS, однако в 2010 году накопила в себе критическую массу отличий и была выведена в отдельную ветвь разработки.

Основная цель создания VQS – избавить разработчиков от создания однотипного примитивного кода (например, блоки, связанные с пользователями – регистрация, авторизация, проверка прав; реализация шаблона MVC для любых расширений системы; интеграция сторонних библиотек – визуального редактора, всевозможных парсеров, captcha и подобных систем) за счёт предоставления его функциональности посредством внутренних библиотек и стандартных расширений системы. При этом важно создать систему, которая отличалась бы простотой и низким порогом вхождения с точки зрения разработчика.

Основной аудиторией системы являются web-разработчики. Изначально VQS ориентирована на создание не сайтов, а web-приложений. Однако важно было так же поддержать основной минимум функциональности, принятой в области web-разработки. Таким образом, система на уровне ядра должна обеспечивать поддержку концепции страниц, ЧПУ, многосайтовость и гибкое разделение прав по группам.

Требования

Таким образом, разрабатываемая система должна:

- По умолчанию создавать сайт, состоящий из страниц
- По умолчанию поддерживать базовые функции работы с пользователями
- По умолчанию поддерживать ЧПУ
- По умолчанию поддерживать возможность интегрировать любую имеющуюся библиотеку

- Реализовывать любые дополнительные требования через расширения
- Обеспечить разработчика всем необходимым, чтобы создать новое расширение или адаптировать имеющуюся библиотеку, в том числе
    - Документацией
    - Примерами
    - API
- Иметь средства повышения надёжности и быстродействия такие как
    - Система контроля изменений страниц
    - Система обнаружения вирусного заражения
    - Система блокировки пользователей и посетителей
    - Система кеширования

Все указанные выше требования были преобразованы в концепции, по которым система и разрабатывалась. По ним же она продолжает развиваться.

Концепция форматтеров-хайлайтеров (highlighters)

Highlighters — это концепция, позволяющая разработчику реализовать сложную логику преобразования какого-то блока текста, а пользователю — легко передать этот блок текста на оформление.

В теле вака-разметки использование хайлайтера выглядит так %%(html) testdiv ...%% — т.е. используется «парный символ» %% с указанием в круглых скобках, какой функции-хайлайтеру передать то, что внутри.

Разработчик же пишет эти самые функции, принимающие текстовую переменную, совершающие с ней какие-то преобразования и затем возвращающие результат.

Как правило, этот синтаксис используется для разнообразной «расцветки» текста — для подсветки синтаксиса, оформления ICQ-логов и email, вставки каких-то произвольных блоков текста.

Эта концепция — способ легко расширить разметку, при этом:

- не задействуя дополнительных «парных символов»;
- не разбираясь в деталях работы вака-форматтера.

Концепция экшнов-действий (actions)

Actions — это концепция, позволяющая разработчику реализовать какую-то сложную функциональность (возможно, даже требующую диалога с пользователем), а пользователю — легко обеспечить в нужном месте документа доступ к этой функциональности.

В теле вака-разметки использование «действия» выглядит так {{RecentChanges for="kuso@npj"}} — используется «парный символ» {{ с указанием имени «действия» и дополнительных параметров.

Разработчик пишет эти функции точно так же, как он пишет и «хайлайтеры».

Именно через «действия» в WackoWiki Very Quick Start реализована значительная часть функциональности по работе с документами (построение каталога, списков последних изменений/комментариев, поиск).

Эта концепция — способ «встроить» в документ какую-то сложную функциональность, при этом так же не влезая в детали работы вака-форматтера.

Концепция расширений (extensions)

Разработчик должен иметь возможность расширять функциональность системы, не изменяя ядро. Такой возможностью должны стать расширения.

Расширения обеспечивают

- внедрение в систему любых новых
    - экшенов
    - форматтеров
    - библиотек
    - шаблонов
-

Обозначения
Здесь и далее:

- ExtensionName* – название расширения
- actionname – название произвольного экшена, включенного в расширение
- handlername – название произвольного хендлера, включенного в расширение
- templatename – имя файла произвольного шаблона, включенного в расширение
- apiname – имя файла произвольного набора функций (api или библиотек), включенного в расширение
- filename – имя произвольного файла
- так обозначаются каталоги
- так – файлы

* Обращаем внимание на то, что экшены, хендлеры, шаблоны и файлы библиотек (api) лучше называть строчными буквами. На названия самих расширений это правило не распространяется.

Общая структура
Максимально-полное дерево каталогов выглядит так:

- ExtensionName
    - ExtensionName.php1
    - actions
        - actionname.php
    - lang

        - wakka.ru.php2
        - wakka.en.php
    - lib
        - apiname.php3 – функции, который будут доступны везде. обращаться к этим функциям напрямую (без $this->), поэтому рекомендуется в названии добавлять префикс с именем расширения (напр. WPTRN_ИмяФункции)
    - handlers
        - handlername
            - show.php4
        - handlername.init.php5
    - smarty
        - cache
        - config
        - templates
            - templatename.txt6
        - templates_c
    - run
        - makejs.php – автоматически-подключаемые js-скрипты
        - smartyplugins.php – вроде почти не используется – по идее выполняется при создании нового Smarty-шаблона у расширений (и то при условии вызова $kernel->smartyServicesPlugins($sm)
        - install.php – действия при установке этого сервиса
        - uninstall.php – действия при удалении этого сервиса
        - performance.php – действия по улучшению производительности (напр. очистить кеш)
        - userinfo.php – выводится в экшене логина – можно показать дополнительную инфу о пользователе (напр. аватар), не меняя стандартный экшн
        - init.php – действия, выполняемые при инициализации ядра (в конструкторе VQSkernel). Многое ещё не инициализировано (напр. проверка прав)
        - stat.php – действия, выполняемые при запуске функции ядра Run – через неё выводятся страницы (из названия ясно, что рекомендуется использовать для отслеживания статистики)

- onrun.php – выполняется при запуске функции Run – к этому моменту всё, что нужно уже инициализировано
- removepage.php – действия при удалении страницы
- afterpagerender.php – действия после выполнения всех команд на странице
- beforepagerender.php – действия перед выполнением всех команд на странице
- infohandler.php – выводится в настройках страницы (напр. «Редактировать с помощью шаблонов»)
- onshow.php – выполняется в методе show (т.е. при обычном показе страниц) на страницах после вывода текста страницы
- themeheader.php – выполняется в themes/standartHeaderWithoutBody.php, т.е. в разделе head дизайна

Концепция виртуальных форм

В VQS введён собственный аналог html-форм – виртуальный формы. Это некая абстракция соответствующих элементов, представляющая собой обёртку с определённой логикой. Такой подход позволяет ускорить процесс разработки собственных интерфейсов. Физически располагаются там же, где и экшны, и могут по аналогии с ними подключаться.

Все виртуальные формы расположены в логической ветке actions/vrfm системы VQSDevelop.

Подробнее о том, что собой представляет каждый элемент виртуальной формы, ниже.

Виртуальные формы используются в smarty и не только. В первую очередь благодаря тому, что в них есть построцессинг в формат safe_html.

Button	Кнопка submit, у которой работают все стили
Int	Элемент выбора целого числа. Выглядит, как текстовое поле ввода с текстом рядом
Select	Выпадающий ComboBox
SaveSelect	То же самое, что Select, только автоматически сохраняет свое значение между перезагрузками страницы
TinyMCE	Визуальное редактирование HTML на основе TinyMCE. Начинает работать после установки расширения TinyMCE, в противном случае представляет текстовое поле
WikiEditor	Текстовая область с возможностью вики-разметки
Text	Простое текстовое поле
SaveText	То же самое, что Text, только автоматически сохраняет свое значение между перезагрузками страницы
TextArea	Текстовая область
SaveTextarea	То же самое, что Textarea, только автоматически сохраняет свое значение между перезагрузками страницы
DateTime	Элемент выбора даты/времени. Выглядит, как текстовое поле ввода с картинкой рядом
Date	Элемент выбора даты. Выглядит, как текстовое поле ввода с картинкой рядом
Hidden	Спрятанное поле. Добавлено для общей совместимости
CheckBox	Чекбокс. Возможные значения: 0 (неустановлен), 1(установлен)
TSelect	Выпадающий ComboBox. Позволяет выбрать элемент таблицы из БД, возвращая его ключ

SelectInt	Select для диапазона чисел от FROM до TO
SelectRadio	Список Radio, позволяющий выбрать одну из них
KNotUserCaptcha	Показывает капчу (lib/kcaptcha/) только для незарегистрированного пользователя, и требует ввода в текстовое поле. Возвращает 1, если капча была распознана верно, 0 – если неверно. Для зарегистрированного пользователя всегда возвращает 1
KCaptcha	Показывает капчу (lib/kcaptcha/) и требует ввода в текстовое поле. Возвращает 1, если капча была распознана верно, 0 – если неверно
ImageHttpLoader	Текстовое поле. Если ввести адрес удаленной картинки, при постпроцессинге закачивает её на сервер и сохраняет как файл, заменяя результат.

Концепция MCV

Система пропитана идеологией шаблона MVC – на каждом этапе разработки – будь то дизайн сайта или создание экшена – разработчика вынуждают использовать в работе шаблон MVC с теми или иными дополнениями.

MVC подход реализован с помощью шаблонизатора Smarty.

Smarty – язык разметки шаблонов и интерпретатор этих шаблонов.

Smarty – это компилирующий обработчик шаблонов для PHP. Говоря более четко, он предоставляет один из инструментов, которые позволяет добиться отделения прикладной логики и данных от представления. Это очень удобно в ситуациях, когда программист и верстальщик шаблона – различные люди.

Предустановленные переменные в VQS

При каждом создании Smarty-объекта движок VQS автоматически устанавливает в нем ряд переменных:

- themeurl – адрес темы. Обычно обработка картинок производится QS автоматически, и перед картинками не надо ставить адрес темы. Но в других случаях это может быть востребовано.
- rooturl – адрес корня инсталляции VQS
- today – сегодняшний день, в формате Y-m-d
- yesterday – вчерашний день, в формате Y-m-d
- isadmin – является ли пользователь администратором

Встроенные в VQS модификаторы

В Very Quick Start доступны следующие дополнительные модификаторы (+ еще можно дописывать собственные):

- date_format_rus – позволяет отформатировать дату на русском языке
- date_format_rus_curbased – форматирует дату на русском языке относительно текущего дня (т.е. вставляет слова «Вчера», «Сегодня»)
- themeblock – вставляет блок сервиса ThemeBlock. Пример: <img src="images/{"customlogo"|themeblock}" alt="logo">

- themeelement – вставляет элемент темы. Пример: {"adminlinks"|themeelement}
- wackoformat – осуществляет форматирование вики-текста. Пример: {"menu"|themeblock|wackoformat}

Функции в VQS

Кроме того, в VQS есть собственные хорошие функции.

- wackohref tag="tag" param="param" method="method" – вставляет ссылку на страницу. Пример: Ответить на вопросы (обратите внимание, как передаются в функцию переменные, зависящие от других Smarty-переменных).
- wackoformopen – открывает форму
- wackoformclose – закрывает форму
- virtualform id="id" name="name" default="def" – вставляет компонент виртуальной формы. Семантика параметров соответствует функции vfrmField.

Тестирование

Описание системы VQSDevelop&Test

На рисунке 1 представлен интерфейс результатов тестирования. Он выводится при просмотре очередной функции.

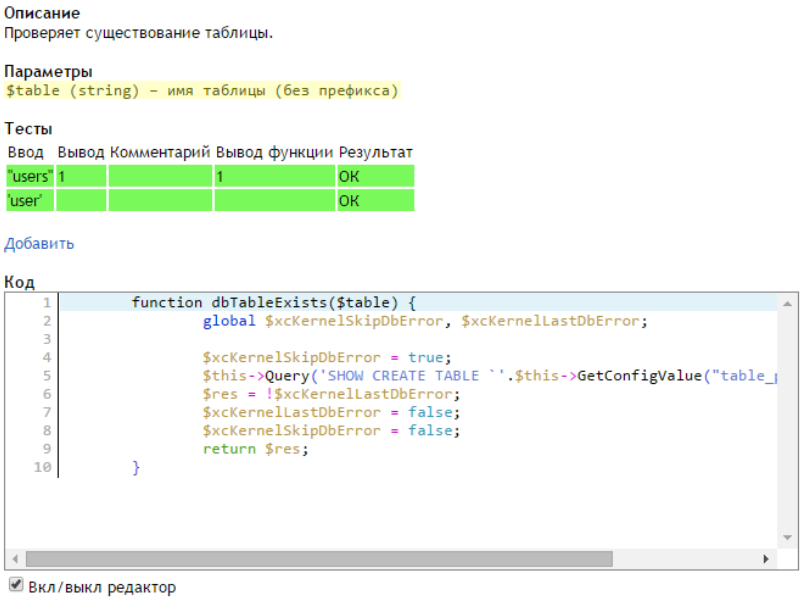

Рис. 1 «Результаты тестирования функции dbTableExists»

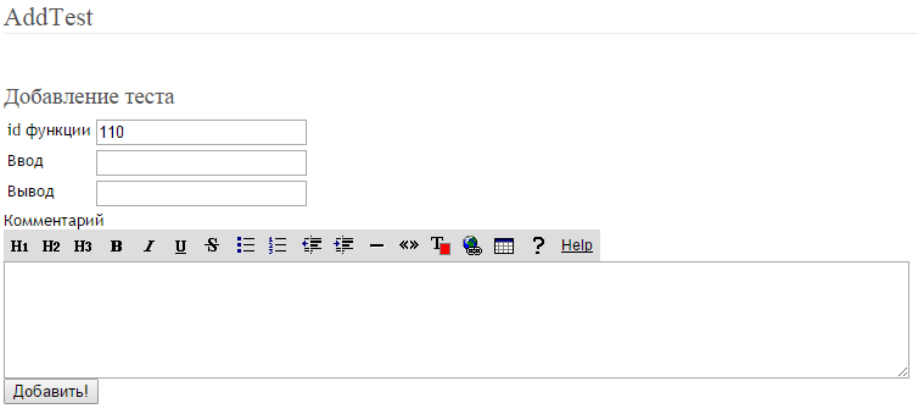

Рис. 2 «Добавление теста»

Добавление тестов осуществляется с помощью интерфейса добавления (см. рис. 4).

Модульные тесты

Модульные тесты осуществляется с помощью средства VQSDevelop посредством исполнения кода в безопасной среде и проверки получаемых

значений. Код выполняется в контексте ядра – к моменту вызова тех или иных функций, система уже запущена, а ядро собрано.

Всего в систему введено около 120 функций. Тесты написаны для 10.

Пример результата тестирования см. на рис. 1.

Интеграционные тесты

Интеграционные тесты реализованы с помощью той же системы VQSDevelop.

Для примера протестирована функция CanAction() (см. рис. 5). Она вызывает функции LoadSingle() и GetConfigValue(). Для этих функций так же написаны тесты (см. рис 3-4).

Таким образом, протестированы функции LoadSingle() и GetConfigValue(). И построенная с использованием этих функций CanAction().

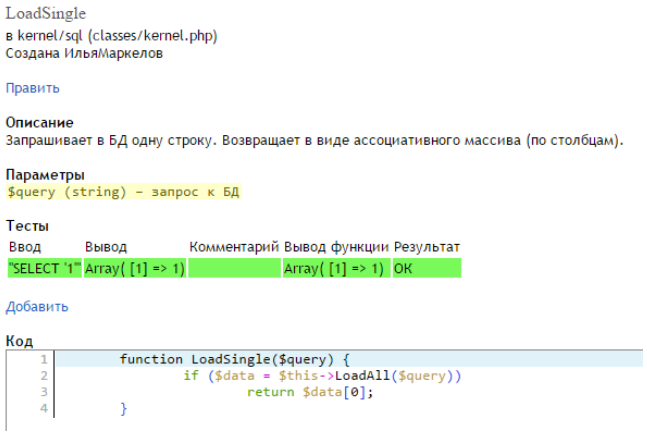
LoadSingle
в kernel/sql (classes/kernel.php)
Создана ИльяМаркелов
Править
Описание
Запрашивает в БД одну строку. Возвращает в виде ассоциативного массива (по столбцам).
Параметры
$query (string) – запрос к БД
Тесты
Ввод | Вывод | Комментарий | Вывод функции | Результат
'SELECT '1'' | Array([1] => 1) | | Array([1] => 1) | OK
Добавить
Код
function LoadSingle($query) {
if ($data = $this->LoadAll($query))
return $data[0];
}

Рис 3 «Результаты тестирования LoadSingle()»

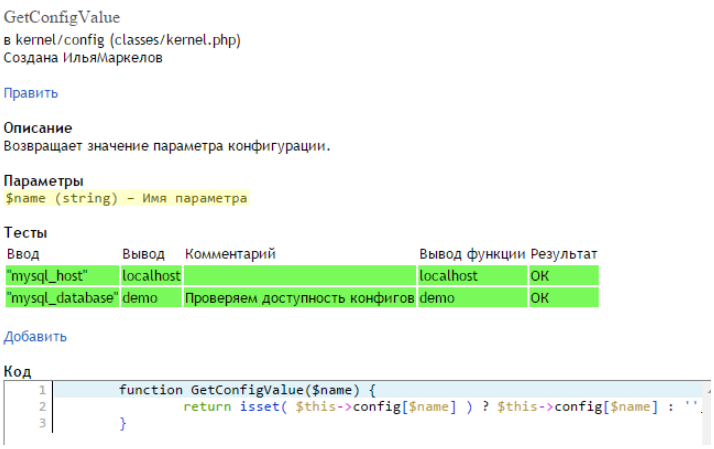
GetConfigValue
в kernel/config (classes/kernel.php)
Создана ИльяМаркелов
Править
Описание
Возвращает значение параметра конфигурации.
Параметры
$name (string) – Имя параметра
Тесты
Ввод | Вывод | Комментарий | Вывод функции | Результат
"mysql_host" | localhost | | localhost | OK
"mysql_database" | demo | Проверяем доступность конфигов | demo | OK
Добавить
Код
function GetConfigValue($name) {
return isset($this->config[$name]) ? $this->config[$name] : '';
}

Рис 4 «Результаты тестирования GetConfogValue()»

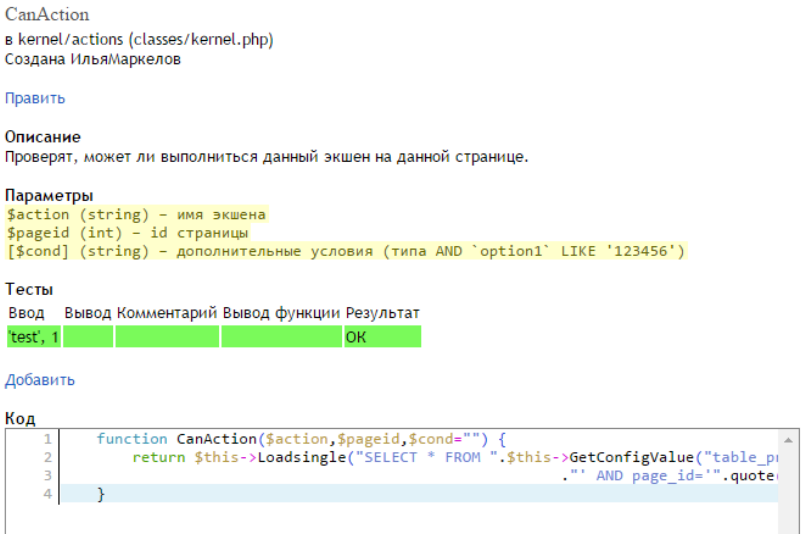

Рис. 5 «Результаты тестирования CanAction()»

Системные тесты

Системное тестирование осуществляется помощью системы Selenium IDE.

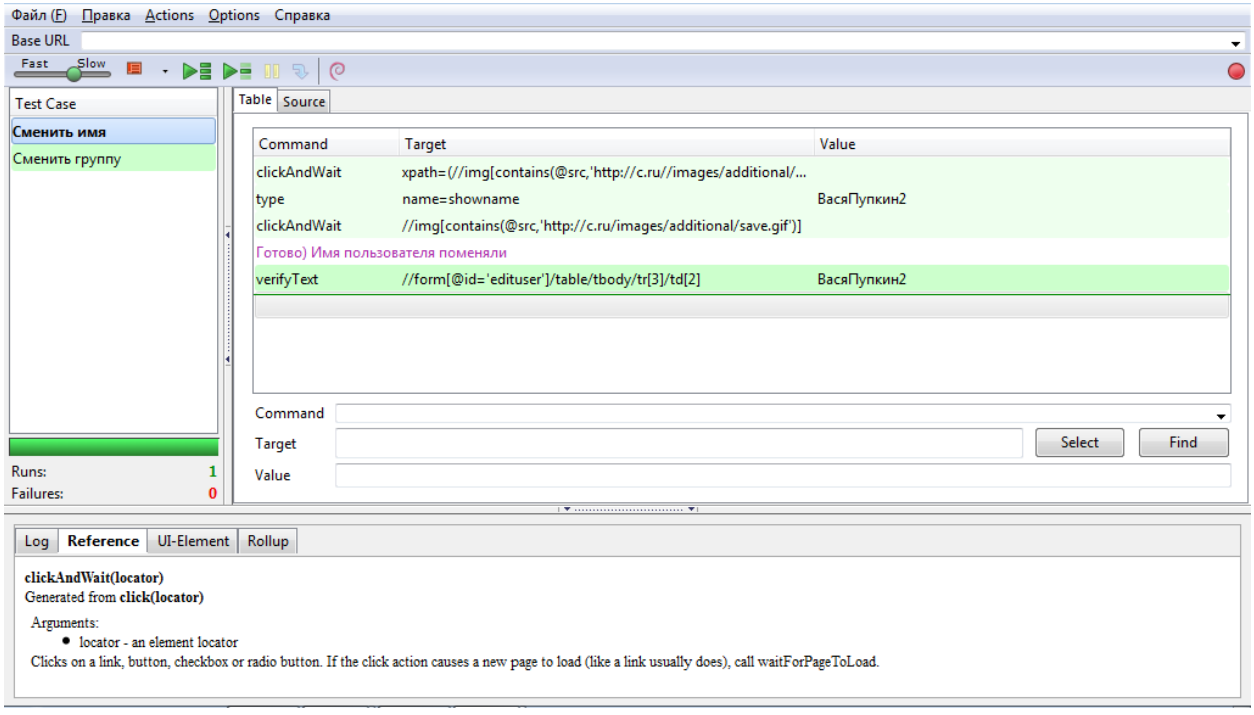

Рис. 6 «Результаты тестирования экшена «UserManager»

Протестирован экшен управления пользователями системы, производящий основные манипуляции с юзерами (см. рис 6) и система редактирования страниц (см рис. 7)

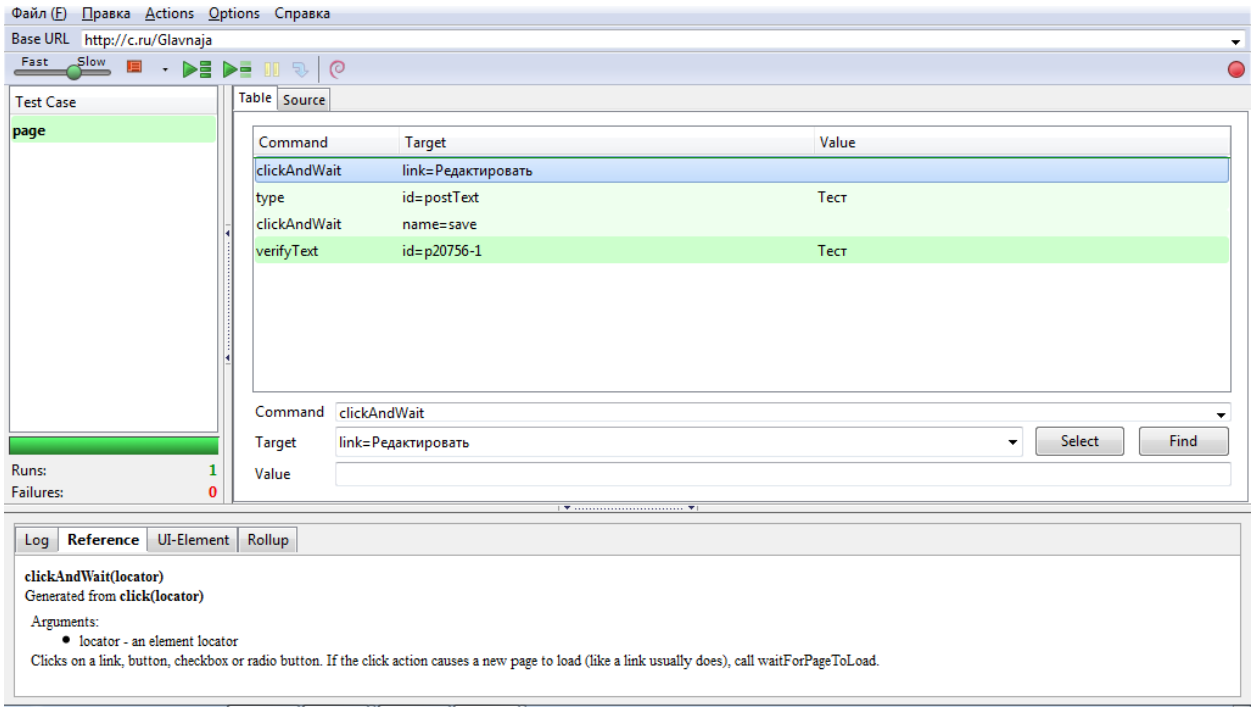

Рис 7 «Редактирование страницы»

В обоих случаях, наборы тестов завершились успешно.

Тесты введены в формате selenium.html. Пример:

page		
clickAndWait	link=Редактировать	
type	id=postText	Тест
clickAndWait	name=save	
verifyText	id=p20756-1	Тест