Методика автоматизированного функционального тестирования

# Общие положения

* 1. Настоящий документ определяет методику и инструментарий автоматизированного функционального тестирования (далее – методика) разрабатываемых и внедренных систем, проводимого Отделом тестирования и внедрения Департамента интеграционных решений ООО «ИБС» (далее - Исполнитель).
  2. Документ предназначен для оценки и планирования затрат на проведение тестирования руководителями проектов и проектной командой.

# Краткое описание

Автоматизированное тестирование программного обеспечения — часть процесса [тестирования](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) на этапе контроля [качества](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%BE_%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%86%D0%B8%D0%B8) в процессе разработки [программного обеспечения](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Оно использует программные средства для выполнения тестов и проверки результатов выполнения, что помогает сократить время тестирования и упростить его процесс.

Существует два основных подхода к автоматизации тестирования: тестирование на уровне кода и GUI-тестирование. К первому типу относится, в частности, [модульное тестирование](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Ко второму — имитация действий пользователя с помощью специальных тестовых [фреймворков](http://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA" \o "Фреймворк).

Автоматизированное тестирование обеспечивает преимущества, которые могут повысить эффективность работы отдела тестирования в долгосрочной перспективе.

Автоматизированное тестирование позволяет:

* проводить чаще регрессионное тестирование
* быстро предоставлять разработчикам отчет о состоянии продукта
* получить потенциально бесконечное число прогонов тестов
* обеспечить поддержку Agile и экстремальным методам разработки
* сохранять строгую документацию тестов
* обнаружить ошибки, которые были пропущены на стадии ручного тестирования

Кроме этого, существуют положительные побочные эффекты от применения автоматизированного функционального тестирования:

* тестируемая ИС в результате тестирования наполняется данными (которые используются последующими итерациями тестирования, в т.ч. и нагрузочным тестированием)
* повышается профессиональный уровень тестировщика, так или иначе участвующего в процессе автоматизации тестирования

Наиболее распространенной формой автоматизации является тестирование приложений через графический пользовательский интерфейс. Популярность такого вида тестирования объясняется двумя факторами: во-первых, приложение тестируется тем же способом, которым его будет использовать человек, во-вторых, можно тестировать приложение, не имея при этом доступа к исходному коду.

Существует 4 поколения инструментов и техник автоматизации тестирования:

* Утилиты записи и воспроизведения (capture/playback tools) записывают действия тестировщика во время ручного тестирования. Они позволяют выполнять тесты без прямого участия человека в течение продолжительного времени, значительно увеличивая продуктивность и устраняя «тупое» повторение однообразных действий во время ручного тестирования. В то же время, любое малое изменение тестируемого ПО требует перезаписи ручных тестов. Это поколение инструментов не эффективно и не масштабируемо.
* Сценарии (Scripting) — форма программирования на языках, специально разработанных для автоматизации тестирования ПО — смягчает многие проблемы capture/playback tools. Но разработкой занимаются программисты высокого уровня, которые работают отдельно от тестировщиков, непосредственно запускающих тесты. К тому же скрипты более всего подходят для тестирования GUI и не могут быть внедренными, пакетными или вообще каким-либо образом объединены в систему. Наконец, изменения в тестируемом ПО требуют сложных изменений в соответствующих скриптах, и поддержка все возрастающей библиотеки тестирующих скриптов становится в конце концов непреодолимой задачей.
* Data-driven testing — методология, которая используется в автоматизации тестирования. Особенностью является то, что тестовые скрипты выполняются и верифицируются на основе данных, которые хранятся в центральном хранилище данных или БД. Роль БД могут выполнять ODBC-ресурсы, csv или xls файлы и т. д. Data-driven testing — это объединение нескольких взаимодействующих тестовых скриптов и их источников данных в фреймворк, используемый в методологии. В этом фреймворке переменные используются как для входных значений, так и для выходных проверочных значений: в тестовом скрипте обычно закодированы навигация по приложению, чтение источников данных, ведение логов тестирования. Таким образом, логика, которая будет выполнена в скрипте, также зависит от данных.
* Keyword-based (driven) testing подразумевает разделение процесса создания кейсов на 2 этапа: этап планирования и разработки фреймворка и этап реализации тестов.

Этап разработки фреймворка предполагает создание карт объектов приложения и набора ключевых слов, с помощью которых будет производиться реализация тестов. Пример простого ключевого слова (одно действие для одного объекта) является ввод «Имени пользователя» в текстовое поле.

|  |  |  |
| --- | --- | --- |
| Объект | Действие | Данные |
| Textfield (username) | Enter text | username |

Пример сложного ключевого слова (комбинация ключевых слов в сложном модуле) логин в систему.

|  |  |  |
| --- | --- | --- |
| Объект | Действие | Данные |
| Textfield (domain) | Enter text | domain |
| Textfield (username) | Enter text | username |
| Textfield (password) | Enter text | password |
| Button (login) | Click | One left click |

Этап реализации отличается в зависимости от используемого фреймворка или инструмента автоматизации. В большинстве случаев специалисты по автоматизации тестирования реализуют фреймворк, который обеспечивает ключевые слова: «Проверить» (check), «Ввести» (enter), «Нажать» (Click) и др.

Специалисты по мануальному тестированию или тест дизайнеры, разрабатывают тесты (test cases) на базе ключевых слов, которые определены и разработаны на стадии планирования. Тесты выполняются на основе драйвера, который вычитывает ключевые слова и выполняет соответствующий код.

**Преимущества**

Не смотря на то, что Keyword-Driven подход требует более длительную стадию планирования и время подготовки до физического старта разработки конкретных тестовых случаев, чем другие методологии автоматизации, в конечном итоге он делает стадию создания и поддержки тестов более эффективной, структура тестов более надежна и может быть подвержена обновлению с минимальными затратами.

Чем более абстрактные ключевые слова использованы, тем легче их применять и тем проще проходит стадия поддержки. Модульная структура Keyword-Driven тестов позволяет создавать новые тесты, используя уже разработанные модули.

# Трудоемкость и экономическая оправданность

Одной из главных проблем автоматизированного тестирования является его трудоемкость: несмотря на то, что оно позволяет устранить часть рутинных операций и ускорить выполнение тестов, большие ресурсы могут тратиться на обновление самих тестов. Это относится к обоим видам автоматизации. При рефакторинге часто бывает необходимо обновить и модульные тесты, и изменение кода тестов может занять столько же времени, сколько и изменение основного кода. С другой стороны, при изменении интерфейса приложения необходимо заново переписать все тесты, которые связаны с обновленными окнами, что при большом количестве тестов может отнять значительные ресурсы.

Не всегда полезно автоматизировать тесты. Иногда ручное тестирование может оказаться более подходящим. Например, если графический интерфейс приложения сильно изменится в ближайшем будущем, автоматизированные тесты придется переписывать. К тому же иногда попросту не хватает времени на автоматизацию. В краткосрочной перспективе ручное тестирование может быть более эффективно. Если приложение должно быть выпущено в очень сжатые сроки, и нет готовых автоматизированных тестов, но протестировать в срок необходимо, то ручное тестирование является лучшим решением.

Условие малой трудоемкости подготовки автотестов и быстрое получение результатов автоматизированного тестирования обычно выполняется при автоматизации следующих видов деятельности тестировщиков / специалистов по сопровождению:

* мониторинг доступности приложения (оценивается не функциональность, а сам факт работы приложения);
* мониторинг доступности сервиса, реализующего взаимодействие между компонентами сложных программных комплексов посредством стандартных протоколов обмена данными (SMTP/POP3, FTP, HTTP и т.д. - по сути, проверяем, например, ходит ли почта между подсистемами А и Б).

Экономическую целесообразность автоматизации можно определить следующей формулой:

NTtPt

A = ---------------

L + TaPa

где

A — коэффициент возврата вложений в автоматизацию;

N — количество выполнений данного набора тестов в течение цикла разработки;

Tt — время, затрачиваемое на выполнение набора тестов вручную, ч.;

Pt — зарплата тестировщика в час;

L — стоимость лицензии(й) программного обеспечения для тестирования;

Ta — время, затраченное на разработку и поддержку автоматизации, ч.;

Pa — зарплата инженера по автоматизации тестирования ПО в час.  
**Результат**

A>1 означает, что автоматизация в данном проекте является экономически оправданной.

Данная формула учитывает лишь экономическую выгоду для отдела тестирования. При расчётах для всего отдела разработки эту формулу нужно переписать следующим образом:  
 NTtPt+Pn(Tt-Tn)

A = --------------------------

L + TaPa

где

Pn — зарплата программиста, в час;

Tn — время, затрачиваемое на автоматический набор тестов, ч.

# Принципы автоматизации тестирования

При планировании систематичной и продуктивной автоматизации тестирования, полезно будет помнить о нижеследующих принципах. Источник- «Test Automation Snake Oil» и «Lessons Learned in Software Testing».

1. Автоматизация тестирования не может заменить тестировщика  
   Человек-тестировщик, даже без опыта и специального образования, способен выполнять такие задания и замечать такие моменты, которые автоматизированный тест либо не сможет реализовать, либо не заметит. Но реалии таковы, что даже со всеми ограничениями, [автоматизация может иметь существенное значение в процессе тестирования](http://automated-testing.info/knowledgebase/article/chto-mozhet-predlozhit-avtomatizacija-testirovanija-ili-vygody-avtomatizacii-t). Зачастую, более эффективно думать про автоматизацию как про расширение возможностей людей-тестировщиков, а не про замену ручного тестирования автоматизированным. Эффективные попытки автоматизации начинаются с эффективного обдумывания процесса тестирования. Отсутствие хорошей стратегии тестирования делает из автоматизации серию повторяющихся действий, которые редко помогают выявить [ошибку](http://automated-testing.info/knowledgebase/article/rasprostranennye-oshibki-avtomatizacii-testirovanija).
2. Автоматизация тестирования – это больше, чем просто выполнение тестов  
   Большинство людей, услышавших об автоматизированном тестировании впервые, представляют себе что-то вроде «тестирования пока мы спим». Другими словами, они просто ожидают, что компьютер запускает тесты. Это действительно один из полезных видов автоматизации. Но автоматизация тестирования подразумевает под собой, помимо этого, намного больше. Например, для каждого пункта в приведенном далее списке, можно применить автоматизацию в какой-то мере, даже если все остальные будут выполняться «вручную».

* [Генератор тестов](http://automated-testing.info/news/generacija-testovyh-dannyh-s-pomoshhju-generatedatacom) (генератор данных и скриптов). Инструменты автоматизации могут создавать специальные данные, например, произвольные электронные письма, или наполнять базу данных, или генерировать комбинации параметров и данных, которые нам необходимы при тестировании;
* Конфигурации системы. Инструменты могут сохранять или воссоздавать параметры системы, устанавливать систему в определенное состояние, или создавать, или восстанавливать «образы» жестких дисков;
* Симуляторы. Инструменты могут симулировать подсистемы или условия среды (окружения), которые не доступны (или временно недоступны) для тестирования, либо слишком дорогие, чтобы использовать их в реальности;
* Выполнение тестов (среда и тестовые скрипты). Инструменты могут управлять тестируемым приложением самостоятельно, или симулировать работу пользователя с графическим интерфейсом приложения, или обходя графический, использовать альтернативный тестируемый интерфейс;
* Исследование. Инструменты могут показать то, что осталось бы невидимым для людей. Они могут статично анализировать продукт, парсить файлы с логами, или проводить мониторинг параметров системы.
* «Детекторы»(оракулы). Под детектором подразумевается любой механизм для определения успеха или неудачи. Инструменты могут выявлять определенные виды сбоев в продукте.
* Запись действий и анализ охвата тестами. Инструменты могут отслеживать процесс тестирования и предоставить отчет о том, что было и что не было бы протестировано. Также могут записывать выполняемые действия, чтобы позже применить их в других тестах.
* Управление процессами тестирования. Инструменты могут сообщать о результатах тестирования, упорядочивать идеи для тестов, подсчитывать метрики.

1. Автоматизация может мгновенно устаревать  
   Программные проекты напрямую используют код, который находится в эксплуатации. Тестовый код – это не код в эксплуатации. Поэтому в типичном проекте по разработке программного обеспечения разрешаются изменения производственного кода, даже если это повлечет за собой [неработоспособность существующего тестового кода](http://automated-testing.info/knowledgebase/article/stoimost-podderzhki-avtomatizacii-testirovanija). Это вполне нормальное, вообще говоря, это целесообразное, экономически обоснованные действия. Если продукт будет изменяться, вы имеете, в основном, всего два пути: инвестировать средства в тестовый код, чтобы он был устойчив к таким изменениям, либо же делать код настолько дешевым и легкокорректируемым, чтобы не беспокоится о том, что он станет неработоспособным в случае изменений продукта.
2. Тестовых инструментов очень много и они очень разнообразны  
   Большинство людей, особенно менеджеров, считают, что [тестовые инструменты](http://automated-testing.info/tools) – это только то, что представлено на рынке как «[инструменты для тестирования](http://automated-testing.info/tools)». [Зачастую они достаточно дорогие](http://automated-testing.info/knowledgebase/article/ocenka-i-vybor-instrumenta-avtomatizacii). Но, на самом деле, почти все можно использовать в качестве тестовых инструментов, и многие утилиты, которые продаются для совсем других целей, могут оказаться очень полезными для тестирования. Некоторые такие утилиты бесплатные, некоторые поставляются в репозиториях для разработчиков. Ресурсы:

* Универсальная библиотека MSDN;
* Все средства для разработки ПО от Microsoft (в них всегда включены полезные утилиты);
* Наборы для исправлений несовместимостей (compatibility toolkit) от Microsoft ([www.microsoft.com](http://www.microsoft.com/));
* Web-основанные ресурсы для тестирования web-приложений (HTML checker, анализаторы доступности);
* Resource Kit от Microsoft (доступен от Microsoft);
* Скриптовые языки (например, Perl, Ruby, TCL) и связанные с ними библиотеки;
* Открытые хранилища программного обеспечения ([http://download.com](http://download.com/));
* Приложения для мониторинга операционной системы ([http://sysinternals.com](http://sysinternals.com/));
* Приложения для тестирования с открытым кодом ([http://opensourcetesting.org](http://opensourcetesting.org/));
* Программы-шпионы для экспериментальных тестов ([www.spectorsoft.com](http://automated-testing.info/node/245/www.spectorsoft.com));
* Случайные разработки (бывает, что кто-то в свободное время взял и уже разработал необходимую утилиту).

1. Автоматизация тестирования зависит от тестируемости продукта  
   Некоторые виды автоматизированных тестов могут быть экономически выгодными, только если тестируемый продукт спроектирован так, чтобы облегчить тестирование, или хотя бы чтоб тестирование не было слишком сложным. Тестируемость продукта базируется на двух основополагающих понятиях: контролируемость (управляемость) и наблюдаемость. Графический интерфейс не такой тестируемый, как, к примеру, командная строка, файл выполнения или программные интерфейсы, потому что в последних лучше контролируемость. Точно таким же образом, наблюдаемость увеличивается, когда файлы логирования, результаты экспериментов, запросы к базам данных, можно увидеть с помощью графического интерфейса.
2. Автоматизация тестирования может отвлечь от хорошего тестирования  
   Иногда при особой [сфокусированности на автоматизации тестирования](http://automated-testing.info/knowledgebase/article/3-scenarija-kogda-avtomatizacija-mozhet-byt-primenima-ili-ne-mozhet" \o "3 сценария когда автоматизация может быть применима (или не может)" \t "_blank) может возникнуть ситуация, в которой разработчики тестов [отдаляются от самой миссии тестирования](http://automated-testing.info/knowledgebase/article/rasprostranennye-oshibki-avtomatizacii-testirovanija). Они разрабатывают множество инструментов и скриптов, которые хорошо выглядят, но дают слишком мало пользы для согласованной тестовой стратегии, которая и имеет смысл для бизнеса.

# Цикл автоматизации тестирования

Цикл очень простой, так как в реальности уже реализованное решение задачи зачастую требует уточнений и дополнений.

1. понимание того, как проходит тестирование;
2. подбор некоторых технологий, которые будут использоваться для тестирования, по мнению тестировщиков;
3. реализация утвержденного решения проблемы менее чем за неделю;
4. повторение предыдущих шагов.

Этот процесс может проходить на основании пяти документов, плюс несколько фоновых задач. Эти документы должны быть легкодоступны, возможно, выложены где-то на сайте. Вот перечень данных документов:

* Список требований. Это список всех новых запросов и требований от заказчиков (тестировщиков);
* Список задач. В этом документе перечислено, какой задачей занимается каждый из разработчиков тестов;
* Список реализованных решений. Это список уже реализованных решений, которыми на данный момент пользуется команда тестирования. Каждый пункт в этом списке должен содержать короткое описание и отчет о позитивном вкладе решения для увеличения продуктивности тестирования;
* Список запросов на поддержку решений. Это список реализованных решений, которые нуждаются в доработке. Он должен состоять из двух частей: критических доработок и улучшений.
* Список проблем/помех. Это список проблем продуктивности тестирования, которые останутся нерешенными, так как для них нужны новые дорогостоящие инструменты, существенные дополнения в функционале для тестируемости, или для их реализации потребуется больше времени.

Фоновые задачи:

* Взаимодействие с тестировщиками для понимания того, как продукт должен быть протестирован;
* Знакомство со спецификой и используемыми технологиями разрабатываемого продукта, для понимания технической стороны тестирования;
* Работа с тестировщиками и менеджерами по тестированию, чтобы найти оптимальные способы улучшить отчетность и продуктивность.

# Метрики продуктивности тестирования / эффективности автоматизации

* **Коэффициент найденных ошибок, подлежащих исправлению** (за исключением тех, которые невозможно воспроизвести, ошибки тестировщиков, не баги, и другие категории, создающие «шум»). На это число влияет множество факторов. Чрезвычайно опасно использовать только эту метрику. Но как бы то ни было, многие значительные улучшения могут быть привнесены в тестирование исходя из результатов этой метрики.
* **Коэффициент найденных скачков.** Это краткосрочный скачок в количестве найденных багов, вследствие внедрения нового вида тестирования. Ключевой фактор, показывающий, где баги не были найдены без применения нового подхода в тестировании.
* **Коэффициент исправлений.** Возрастающее отношение исправленных багов к найденным говорит о качестве ошибок, сроках, необходимых для сообщения об ошибках, о качестве взаимодействия программистов и тестировщиков. Все это может быть упрощено с применением автоматизации тестирования.
* **Возможности тестирования.** Благодаря автоматизации, можно применить некоторые методы тестирования, которые людям-тестировщикам не по силам. На самом деле эта метрика является списком, который разработчики автоматизированных тестов поддерживают актуальным для периодического пересмотра менеджерами. Вопросы, которые должны задать менеджеры, это «Наступил ли момент использовать эту возможность тестирования?» и «Принесет ли реализация этой возможности тестирования пользу, такую как увеличение уверенности в качестве или коэффициента найденных ошибок».
* **Предотвращение катастроф**. Это список серьезных проблем, которые были предотвращены благодаря автоматизации тестирования. Также может содержать описание предполагаемых действий для тестирования предотвращения повторения обнаруженных раннее критических проблем.
* **Время цикла тестирования**. Сколько времени на него потребуется, если не будут обнаружены критические ошибки, для получения очевидных доказательств надежности определенного релиза продукта? Сокращение этого времени вносит гибкость во весь процесс разработки.
* **Время цикла разработки/тестирования**. На практике, тестирование зависимо от разработки. Мы хотим видеть уменьшение затраченного времени на разработку нового продукта относительно количества задействованного в проекте персонала.
* **Уверенность менеджмента**. Это не совсем метрика, скорее, фактор. Миссией тестирования является обеспечение менеджмента информацией, необходимой ему для принятий бизнес-решений. Видит ли менеджмент улучшения качества по информации, полученной от тестировщиков? Иногда сравнение состояний «до» и «после» позволяют поставить точку.
* [**Стоимость технической поддержки**](http://automated-testing.info/knowledgebase/article/stoimost-podderzhki-avtomatizacii-testirovanija)**.** В конечном счете, ключевым фактором в оценивании успешности тестирования является проверка того, насколько хорошо наше априорное предположение о качестве релиза сравнимо с реальными результатами. Именно поэтому техническая поддержка очень важна. Необходимо внедрить систему регулярно обращаться к сотрудникам технической поддержки и выяснять, какие проблемы возникают у пользователей. Посмотрите на количество звонков в поддержку относительно количества пользователей. Посмотрите, как можно уменьшить время звонка, если предоставить технической поддержке более хорошие инструменты для диагностики, похожие на те, что могут предложить разработчики автоматизированных тестов.
* **Удержание пользователей**. Если качество будет лучше, возможно, пользователи будут довольны.
* **Репутация.** Она может казаться неосязаемой, но это не так. Репутация выражается в историях, рассказываемых о группе тестирования среди других сотрудников компании. Репутация передается в историях, поэтому вы должны иметь видимые, позитивные и впечатляющие победы настолько часто, насколько это возможно. Также, чем чаще команда автоматизации и команда тестирования говорят «да» просьбам о помощи, тем лучше будет их репутация.
* **Завершенные решения**. Ищите способы постоянного увеличения количества завершенных решений задач, выполняемых командой автоматизации тестирования.

# Риски автоматизации тестирования

* автоматизация может не принести существенных улучшений в продуктивности тестирования, если тестировщики не знают, как надо тестировать;
* метрики продуктивности такие как количество созданных или выполненных тест-кейсов за день могут быть очень обманчивыми, и могут привести к большим затратам на проведение бесполезных тестов;
* тестировщики могут не хотеть сотрудничать с разработчиками автоматизированных тестов. Члены команды автоматизации должны быть своего рода консультантами: доступными, отзывчивыми и изобретательными, иначе эта система быстро разрушится;
* разработчики автоматизированных тестов могут предлагать и реализовывать решения тестовых задач, которые постоянно требуют дорогостоящей поддержки;
* разработчикам автоматизированных тестов может не хватать квалификации для планирования и реализации эффективных решений;
* разработчики автоматизированных тестов могут быть настолько успешными, что им не хватает важных заданий для работы и они переключаются на незначительные проблемы
* ошибки в определении целесообразности применения автоматизации функционального тестирования на конкретном проекте могут привести к перерасходу средств, выделенных на тестирование в целом.

# Действия, необходимые для перехода от ручного к автоматизированному тестированию

* Определить и огласить роль команды по автоматизации тестирования разработчикам и тестировщикам;
* Определить [систему отчетности](http://automated-testing.info/knowledgebase/article/ispolzovanie-metrik-v-avtomatizacii-testirovanija-chast-1) и систему координации для команды по автоматизации тестирования;
* Провести [инвентаризацию известных инструментов](http://automated-testing.info/tools) для тестирования в рамках компании;
* Провести [быстрый осмотр инструментов для тестирования](http://automated-testing.info/tools), доступных вне компании;
* Выбрать инструмент тестирования исходя из особенностей архитектуры (платформы и тп) тестируемой системы, бюджета проекта, квалификации специалистов по тестированию и целей тестирования, определенных для конкретного проекта.
* Расположить разработчиков тестов рядом с тестировщиками, чтобы они могли легко разобраться, как происходит тестирование и какие есть возможности для автоматизации тестирования;
* Обучить тестировщиков создавать запросы и оценивать быстро предоставляемые  решения;
* Оценивать прогресс от автоматизации тестирования не реже, чем раз в две недели;
* Продолжать обучение тестировщиков, чтобы они были способны лучше представлять себе усовершенствованные стратегии тестирования.

# Проектирование тестов

**Разновидности тестов**

1. **Тестирование статичного контента**

Самый простой вид тестирования - “контентное”. Это простой тест, который проверяет существование статичного, т. е. не изменяющегося элемента пользовательского интерфейса. Например:

* Правильно ли озаглавлены страницы? Эту проверку можно использовать для того, чтобы удостовериться в том, что тест обнаружил искомую страницу после перехода по ссылке.
* Содержит ли главная страница нужную картинку сверху?
* Расположена ли контактная информация, описание политики конфиденциальности и информация о торговой марке в подвале страницы?
* Стоит ли в начале каждой страницы заголовок с тэгом <h1> Содержит ли этот заголовок верный текст?

Тестировать контент нужно не всегда. Например, если содержание вашей страницы вряд ли будет меняться со временем, то эффективнее проверять его вручную. Если же ваше приложение, например, предполагает перемещение файлов, то контентное тестирование может быть оправдано.

1. **Тестирование ссылок**

Битые ссылки являются частым источником ошибок на веб сайтах. Тестирование подразумевает переход по каждой ссылке и проверку того, что выдана ожидаемая страница. Если статичные ссылки редко меняются, то достаточно будет ручного тестирования. Однако если ваши веб-дизайнеры часто переделывают ссылки, или же файлы периодически перемещаются, то тестирование ссылок следует автоматизировать.

1. **Тестирование функциональности**

Эти тесты направлены на проверку определенного функционала вашего приложения путем передачи ему пользовательского ввода и получения некоторого результата. Очень часто такой тест может состоять из переходов по нескольким страницам, ввода данных в различные формы и поля, операций подтверждения и отмены действий и проверки результатов. Пользовательский ввод может осуществляться с помощью текстовых полей, выпадающих списков, чек-боксов и других форм ввода, которые поддерживает браузер.

Тестирование функциональности - чаще всего самый сложный вид тестирования, который вам придется автоматизировать, но он же и самый важный. Обычно такими тестами выполняется проверка процессов регистрации на сайте, входа в систему, операций над учетной записью пользователя, изменения настроек, сложных операций по получению данных, и тому подобного. В большинстве случаев функциональный тест копирует сценарии использования, определяющие функционал и устройство вашего приложения.

1. **Тестирование динамических элементов**

Зачастую веб-элемент обладает уникальным идентификатором, который дает возможность найти этот элемент на странице. Обычно такие идентификаторы реализованы с помощью HTML атрибутов ‘id’ или ‘name’. Эти значения могут быть как статичными — т. е. неизменяемыми строковыми константами, так и динамически генерируемыми значениями, отличающимися на разных страницах сайта. Например, некий веб-сервер может отображать имя документа как “doc3861” на одной странице, и как “doc6148” на другой, в зависимости от того, какой документ запрашивает пользователь. Это означает, что вашему тестовому скрипту, проверяющему, был ли документ создан правильно, может быть недоступен постоянный соответствующий документу идентификатор. Зачастую страница с какой-либо выдачей, зависящей от действий пользователя, содержит динамические элементы с меняющимися идентификаторами. Впрочем, конкретная реализация, безусловно, зависит от работы веб-приложения.

Например:

**<input type="checkbox" value="true" id="addForm:\_ID74:\_ID75:0:\_ID79:0:**

**checkBox"/>**

Здесь приведен HTML-код для чек-бокса, id которого (addForm:\_ID74:\_ID75:0:\_ID79:0:checkBox), является динамически генерируемым значением. Когда эту же страницу откроют в следующий раз, значение id, вероятно, будет другим.

1. Тестирование AJAX

AJAX - это технология, поддерживающая динамическое изменение элементов пользовательского интерфейса без перезагрузки страницы. Это, например, может быть анимацией, RSS-лентой или обновлением данных в режиме реального времени. Существует несметное количество способов использования AJAX для обновления веб-элемента на странице. Проще всего это можно описать следующим образом: в AJAX-ориентированных приложениях данные, передаваемые сервером, затем отображаются на странице без ее перезагрузки, обновляется только ее часть или же непосредственно измененный элемент.

# Инструменты автоматизированного тестирования

Автоматизированное тестирование обладает множеством достоинств, связанных главным образом с высокой скоростью выполнения тестов и возможностью выполнять однотипные тесты снова и снова. Существует большое количество как коммерческих, так и бесплатных инструментов, помогающих в разработке автоматизированных тестов.

Могут применяться проприетарные инструменты

* HP QuickTest Professional;
* IBM Rational Functional Tester;
* Oracle Application Testing Suite;
* TestComplete;
* Visual Studio Test Professional;
* и открытые инструменты;
* Selenium;
* HTTP Test Tool;
* Watir.

В зависимости от состава команды тестировщиков, их оптыта, требований заказчика выбирается инструмент автоматизированного тестирования. По умоланию первый выбор – Selenium.

Selenium - это комплект из нескольких инструментов, каждый из которых предполагает свой собственный подход к автоматизации тестирования. Большинство инженеров-тестировщиков, которые работают с Selenium, фокусируются на одном-двух инструменах из этого набора, которые лучше всего отвечают их нуждам. Однако изучение всех имеющихся в распоряжении инструментов позволит вам лучше понять все доступные возможности для решения проблем, которые возникают при автоматизации тестов. В совокупности набор инструментов Selenium предоставляет богатый набор возможностей, специально собранных вместе для тестирования всех типов веб-приложений. Selenium предоставляет несколько вариантов для идентификации элементов интерфейса, сравнения ожидаемого и наблюдаемого поведения тестируемого приложения. Одной из ключевых особенностей Selenium является возможность запуска одних и тех же тестов в различных браузерах.

Selenium состоит из нескольких инструментов, каждый из которых имеет свое назначение.

1. **Selenium 2 (или Selenium Webdriver)**

Selenium 2 - последнее пополнение в пакете инструментов Selenium и является основным вектором развития проекта. Это абсолютно новый инструмент автоматизации, который обеспечивает отличный набор возможностей для управления браузером, имеет более целостный и объектно-ориентированный программный интерфейс (API), а также не имеет ограничений, свойственных более ранним версиям.Он появился как результат слияния WebDriver и Selenium пришли к выводу, что каждый продукт обладает своими достоинствами, и слияние двух проектов позволит получить гораздо более надежный инструмент автоматизации.

Результатом этого слияния стал Selenium 2. Этот инструмент предоставляет для использования WebDriver API, а в качестве ядра может использоваться как более новая собственная реализация WebDriver, так и реализация, которая лежала в основе Selenium 1. Кроме того, в Selenium 2 встроен интерфейс Selenium RC для обеспечения обратной совместимости. Это, в частности, позволяет без особых усилий мигрировать существующие тесты на новую версию.

1. **Selenium 1 (или Selenium Remote Control)**

Как вы уже знаете из главы [*Краткая история проекта Selenium*](http://selenium2.ru/docs/introduction.html#selenium-history), на протяжении длительного времени Selenium RC был основным направлением развития проекта, пока в результате слияния WebDriver и Selenium не появился Selenium 2, более новый и мощный инструмент.

Selenium 1 все еще активно поддерживается (в основном в режиме сопровождения), потому что он предоставляет некоторые возможности, которые все еще недоступны в Selenium 2, включая поддержку некоторых языков программирования (например, Perl) и поддержку всевозможных браузеров.

1. **Selenium IDE**

Selenium IDE (Integrated Development Environment, встроенная среда разработки) - инструмент для разработки и создания прототипов тестовых сценариев. Это плагин для браузера Firefox, с простым и удобным интерфейсом для создания автоматизированных тестов. В Selenium IDE встроена функция записи, которая позволяет записывать действия, совершаемые пользователем, и затем сохранять их в виде кода на одном из языков программирования, поддерживаемых Selenium.

1. **Selenium Grid**

Selenium Grid позволяет масштабировать большие тестовые наборы, а также запускать тесты которые необходимо выполнить в нескольких окружениях. Selenium Grid позволяет запускать тесты параллельно, то есть разные тесты могут быть запущены в одно и то же время на нескольких удаленных машинах. Это сулит два преимущества. Во-первых, если у вас очень много тестов или время выполнения ваших тестов слишком велико, вы можете значительно увеличить производительность с помощью Selenium Grid, разделив тесты на несколько потоков и запуская их одновременно на нескольких серверах. Во-вторых, если тесты необходимо запускать в разных окружениях (например в разных браузерах или операционных системах), вы можете настроить удаленные сервера соответствующим образом и запустить одни и те же тесты сразу в нескольких различных средах. В обоих случаях использование параллельных процессов в Selenium Grid позволяет значительно ускорить тестирование.

Приложение 1 – сравнение инструментов автоматизированного тестирования

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№ п/п** | **Критерии сравнения** | ***Automated QA Test Complete*** | ***Mercury QTP*** | ***Rational Functional Tester*** | ***Rational Robot*** | ***Borland SilkTest*** |
|
| ***1.*** | ***Поддрежка различных ОС*** | Windows 98, Windows 2000, Windows Server 2003, Windows Me, Windows NT, Windows XP, Windows Vista | Windows 2000, Windows Server 2003, Windows XP, Windows Vista | Windows 2000, Windows 2003, Windows XP Professional, Windows Vista, Red Hat Enterprise Linux 4.0, SUSE Linux Enterprise Server 9.0 | Windows 98, Windows 2000, Windows 2003, Windows Me, Windows NT, Windows XP | Windows 98, Windows 2000, Windows Server 2003, Windows Me, Windows NT, Windows XP, Windows Vista, Red Hat® Enterprise Linux® WS 2.1 or 3.0 Sun® Solaris™ 9/10 |
| ***2.*** | ***Поддержка различных типов тестирования*** | Функциональное Тестирование. Регрессионное Тестирование.  Нагрузочное Тестирование (Enterprise version). Стресс Тестирование. | Функциональное Тестирование. Регрессионное Тестирование. | Функциональное Тестирование. Регрессионное Тестирование. | Функциональное Тестирование. Регрессионное Тестирование.  Нагрузочное тестирование. | Функциональное Тестирование. Регрессионное Тестирование. Нагрузочное Тестирование (частично). |
| ***3.*** | ***Работа с серверами терминалов*** | **+** | **+** | Поддерживает тестирование терминальных приложений для серверов 3270 (zSeries) и 5250 (iSeries) с помощью модуля IBM Rational Functional Tester Extension for Terminal-based Applications. | Есть доступ к Rational но нет к Rational Administrator и Test Manager +/- | **+** |
| ***4.*** | ***Скриптовый язык; Отладчик; Редактор*** | VBScript, JScript, DelphiScript, C++Script, C#Script Отладчик - интерпритируемый. Редактор - syntax highlighting, outlining, code completion, code templates, bookmarks. | VBScript  Отладчик - интерпритируемый. Редактор -своя IDE, возможно использование сторонней IDE к QTP от Patterson-consulting - Test Disign Studio | Java, Visual Basic .NET | SQA Basic | 4Test Отладчик - подсветка синтаксиса, дополнение кода. Своя оболочка, возможна правка скриптов и фреймов любым редактором. |
| ***5.*** | ***Привязка к интерфесу  Чем больше инструмент привязан к интерфейсу программы, тем больше требуется времени на корректировку скрипта*** | Привязан | Зависит от того, как настроить инструмент. При изменениях интерфейса измениния требуются только для репозитория. | Привязан | Привязан | Скрипты не зависимы от интерфейса.  Фрейм - зависим. |
| ***6.*** | ***Поддержка объктов различных разработчиков (ActiveX, Flash, Java)*** | ActiveX Java Flash | ActiveX Java Flash | ActiveX Java Flash | ActiveX - установив дополнительную компоненту Flash - не поддерживает | ActiveX Java Flash - необходимо дописывать свои функции и подключаемые модули (возможно реализовано в новых версиях). |
| ***7.*** | ***Использование источников внешних тестовых данных (txt, xml...)*** | **+** | **+** | **+** | **+** | **+** |
| ***8.*** | ***Зависимость от програмной и аппаратной конфигурации (чем меньше зависимость, тем лучше)*** | - | - | - | - | - |
| ***9.*** | ***Верификация свойств объектов (Verification Point, Check Points)*** | Производится при помощи меню. Возможность писать вручную. | Производится при помощи меню. Возможность писать вручную. | Производится при помощи меню. Возможность писать вручную. | Производится при помощи меню. Возможность писать вручную. | Возможность писать вручную. |
| ***10.*** | ***Отображение результатов тестирования*** | Отчёты добавляются к проекту. Формируется дерево зависимости скриптов. | Отчеты сохраняются в отведенной папке. Можно задать выведение отчета после прохождения теста. | В отчете каждый шаг описан - пройден или нет. Ссылка с шага ведет на более подробную информацию. | В отчете каждый шаг описан - пройден или нет. Ссылка с шага ведет на более подробную информацию. | Возможность настройки вида отчёта в зависимости от потребностей. (написание дополнительного функционала вручную) |
| ***11.*** | ***Интергация с другими инструментами*** | Visual Studio 2005 Team System | WinRunner Quality Center | Rational TestManager Rational ClearQuest Rational Administrator | Rational TestManager Rational ClearQuest Rational Administrator | SilkCentralTestManager SilkPerfomer StarTeam |
| ***12.*** | ***Тех. Поддержка*** | Полная (инструмент разрабатывается, документация обновляется). | [Полная (инструмент разработывается, документация обновляется). Кастомеры помогают друг-другу, консультации по тел., эл. почте, документация онлайн.](http://support.mercury.com/cgi-bin/portal/CSO/index.jsp) | Полная (инструмент разработывается, документация обновляется). | Не полная (документация не обновляется -последнее 2007-10-08, инструмент не разрабатывается). | Полная (инструмент разработывается, документация обновляется). |
| ***13.*** | ***Обучение*** | [Обучение проводится в Лас-Вегасе](http://www.automatedqa.com/products/testcomplete/tc_summit.asp%20(Las%20Vegas),%20Webinar%20(онлайн%20консультации)) | [Обучение и сертификация онлайн, стоимость 60 - 124 у.е.](http://www.merc-training.com/main/US/news.cfm?NewsID=113&sitepick=US) | Обучение проводится в Москве (АйТи, Аплана). | Обучение проводится в Москве (АйТи, Аплана). | Онлайн треннинги (собирается команда - раздаются задания - эмитация работы в проекте), Тренинги на базе производителя. |
| ***14.*** | ***Навыки програмирования пользователя*** | **+** | **+** | **+** | **+** | **+** |
| ***15.*** | ***Цена*** | [Standart - 999$ Enterprise - 1999$](http://www.automatedqa.com/products/testcomplete/ordertestcomplete.asp) | [Можно купить только пакет - Mercury Functional Testing (QTP+Winranner+Add-Ins) - 9000 euro (1 license)](http://it4business.ru/forum/index.php?showtopic=7691) | IBM Rational Functional Tester Authorized User License + SW Maintenance 12 Months 5440$   IBM Rational Functional Tester Floating User License + SW Maintenance 12 Months 10500$    IBM Rational Functional Tester Authorized User Initial Fixed Term License + SW Maintenance 12 Months 2940$   IBM Rational Functional Tester Extension Floating User License + SW Maintenance 12 Months 2580$ | IBM Rational Robot Authorized User License + SW Maintenance 12 Months (D53NDLL) -4,240.00, IBM Rational Robot Floating User License + SW Maintenance 12 Months (D52ZFLL) - 8,170.00 | Последняя версия около 9000$ |
| ***16.*** | ***Обучаемость*** | Требует изучения языка программирования, который используется инструментом | Требует изучения языка программирования, который используется инструментом | Требует изучения языка программирования, который используется инструментом | Требует изучения языка программирования, который используется инструментом | Требует изучения языка программирования, который используется инструментом |
| ***17.*** | ***Представительсто компании разработчика в странах ближнего зарубежья*** | Россия (Москва) | Россия (Москва) | Россия (Москва) Украина (Киев, Днепропетровск, Полтава) | Россия (Москва) Украина (Киев, Днепропетровск, Полтава) | Россия |
| ***18.*** | ***Поддерживаемые технологии и средства разработки*** | [.NET, Java, Visual C++, Visual Basic, WPF (XAML), Delphi, C++Builder and web applications](http://automatedqa.com/products/testcomplete/supported_dev_tools.asp) | .NET; Java; Oracle 11i; PeopleSoft; Siebel 7; SAP; Terminal Emulators | VS.NET Winforms, J2SE/J2EE, HTML/DHTML, XML, JavaScript и апплетов Java, а также поддерживает библиотеку SWT Java, связанную с оболочкой Eclipse. | Поддерживает разнообразные технологии создания пользовательского интерфейса - от Java- и Web-технологий до всех элементов управления VS.NET, включая VB.NET, J#, C# и Managed C++. | Browser GUIs – DHTML (JavaScript and CSS) – XML data (XML/XSL Web pages) – Microsoft HTC/HTA – Java Applets n Java GUIs – Eclipse 3.0, 3.1 and 3.1 SWT Standalone and Rich Client Platform (RCP) applications – Java AWT – Java Foundation Classes (JFC, Swing) – Symantic Visual Cafe´ n Java – Sun JVM 1.4, 1.5 (Playback on Linux, Solaris and HP-UX via SilkBean) – Microsoft JVM – Sun Plug-in n .NET GUIs – Microsoft .NET WinForms – Infragistics NetAdvantage WinForms (in Forms UltraWinGrid and UltraWin Toolbar controls) 5.2, 5.3 and 6.1 n .NET – Microsoft .NET CLR 1.x, 2.0 n Native GUIs – VisualBasic 6/Active X – PowerBuilder 9.0, 10.0, 10.2 and 10.5 – Win32, MFC – Motif (on Solaris and Linux) – SAP GUI 6.2 and higher (with SilkPerformer for SAP GUI functional testing) – Custom GUI objects via SilkTest Extension Kit n Internalization – Full Unicode enabled (excluding PowerBuilder, Motif and SilkBean) |
| ***19.*** | ***Особенные характеристики - преимущества*** | Удобный интерфейс, меню и работа с объектами/кодом реализована для быстрой разработки. | Дерево объектов. Экран снэпшотов. | Поддержка большого количества технологий.  Распознавание флеш объектов (Version 6.0 release 65 or later Version 6.0 release 69 or later) |  | Инструмент способен работать со всеми типами объектов, которые находятся в его области видимости. Незнакомые объекты определяются в виде CustomWin - работа с ними возможна при написании дополнительных функций. |
| ***20*** | ***Особенные характеристики - недостатки*** |  | Multiple Document Interface отсуттвует - нет возможности открыть более одного теста. | Перегруженный графическими элементами интерфейс. Высокие системные требования: Processor Minimum: 1.5 GHz Intel(R) Pentium(R) 4  Memory Minimum: 1 GB RAM | Инструмент не разрабатывается. |  |
| ***21.*** | ***Браузеры, с которыми работает инструмент.*** | Microsoft Internet Explorer 5, 6, 7 Mozilla Firefox 1.5-2.0 Netscape Navigator ver. 8.1.2 (ограниченная поддержка) | Microsoft Internet Explorer 6.0 Service Pack 1Microsoft Internet Explorer 7.0 Mozilla Firefox 1.5, 2.0 Alpha 3 (Alpha-level support for Bon Echo 2.0a3) Netscape Browser 8.0 | Microsoft Internet Explorer 6.0 (с Service Pack 1) Mozilla Firefox (1.6 или 1.7) |  | Microsoft Internet Explorer 6, 7Mozilla Firefox 1.5NetScape Navigator: 7, 7.1AOL 9, 9 Security Edition |
| ***22.*** | ***Документация на русском языке*** | нет | нет | нет | Нет | Да |

Инструкция по использованию Selenium IDE

[Установка 2](#_Toc236464990)

[Описание интерфейса Selenium IDE 5](#_Toc236464991)

[Пример использования Selenium IDE 6](#_Toc236464992)

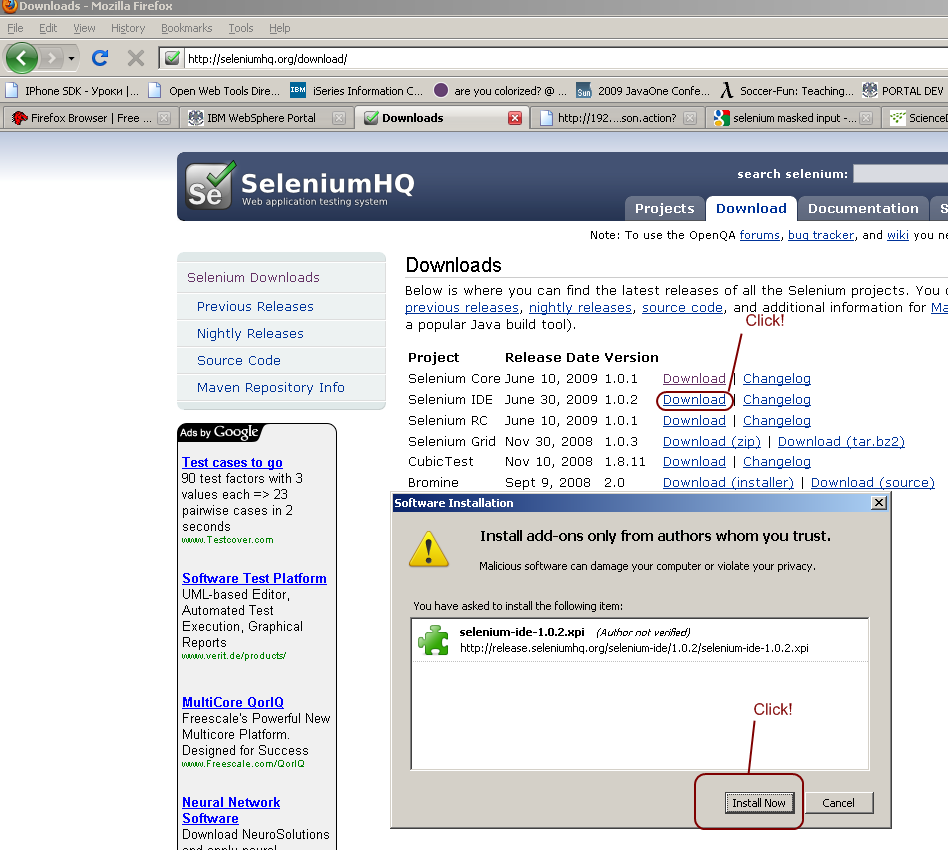
Приложение 2. Использование Selenium для автоматизированного тестирования веб-интерфейсов.

# Установка

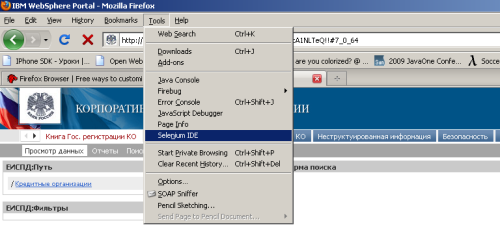
1. Установить [браузер Firefox](http://www.mozilla.com/en-US/firefox/personal.html?from=getfirefox)



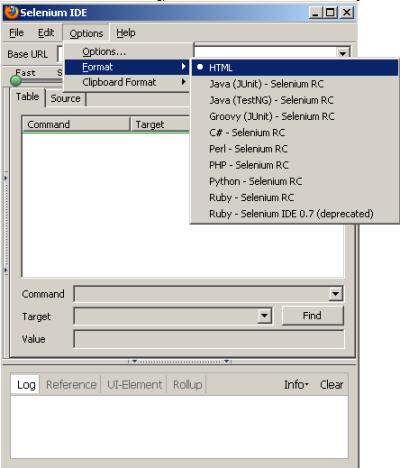
1. Установить [Selenium IDE](http://seleniumhq.org/download/)



1. Закрыть и открыть заново браузер Firefox
2. Запустить Selenium IDE



5. Проверить настройки (установить формат HTML)



# Описание интерфейса Selenium IDE

Все важные кнопки находятся на одной панели:

_images/chapt3_img05_IDE_features.png

*Speed Control, контроль скорости*: Отрегулируйте скорость выполнения теста, используя ползунок. Влево – самая высокая скорость заполнения, вправо – самая медленная.

speed control

*Run All, запустить всё*: Запустить все загруженные в программу тесты.

run all

*Run, запустить только выбранный тест*. Чаще всего, у вас будет только один тест, который вы будете открывать и запускать.

run

*Pause/Resume, пауза/продолжение* выполнения теста. В любой момент вы можете поставить на паузу выполнение теста, а затем продолжить его с места остановки.

pause resume

*Step, шаг за шагом*. Каждый тест состоит из команд. При помощи этой кнопки, вы можете шаг за шагом переходить от одной команды к другой. Редко используется.

step

Эти кнопки вы использовать не будете.

testrunner, rollup

*Record, записывать действия*. Самая важная кнопка.

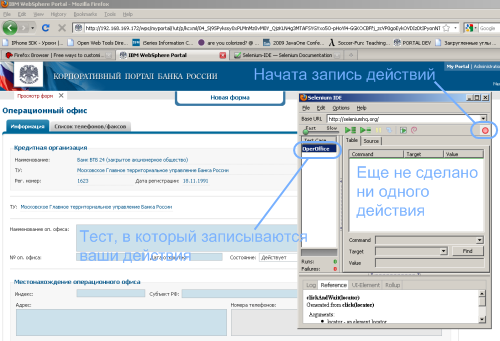
record

# Пример использования Selenium IDE

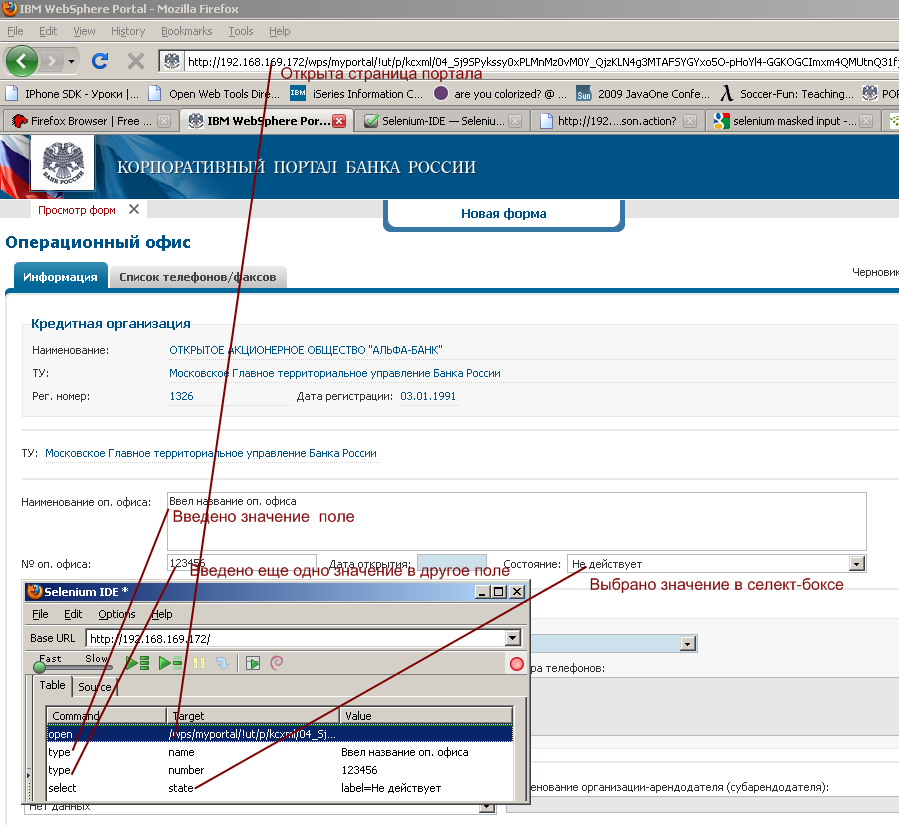
1. Запустите Selenium IDE

2. Откройте нужную страницу, подойдет любая форма КГР.

3. Нажмите на кнопку record, теперь все ваши действия будут записаны.



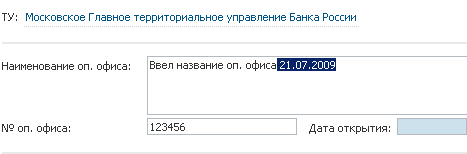
4. Заполняйте форму.



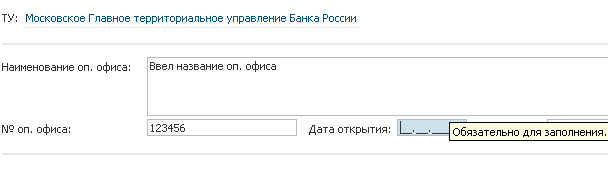
1. Ввод даты в поле.

В связи с тем, что мы используем хитрый плагин для ввода даты в соответствии с маской: дд.ММ.гггг, дату нужно вводить следующим образом.

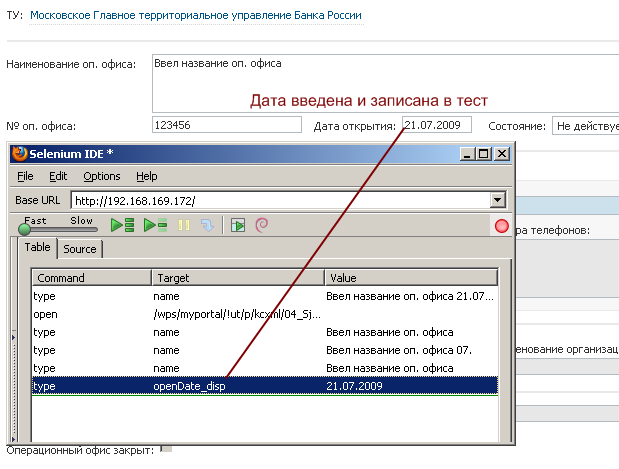
5.1. Набрать дату в каком-нибудь окошке:



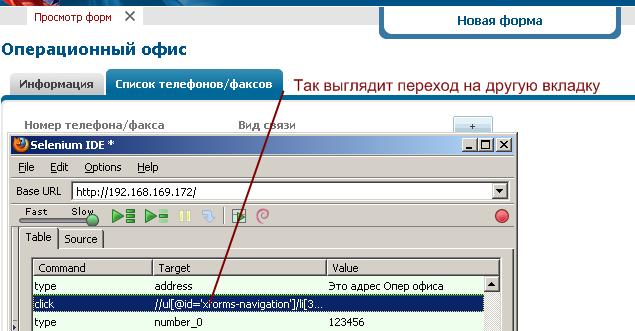
* 1. Нажмите ctrl+x, чтобы «вырезать» дату из окошка и скопировать её в буфер обмена.
  2. Кликните в поле ввода даты, которое вы хотите записать значение, сохранённое в буфере обмена.



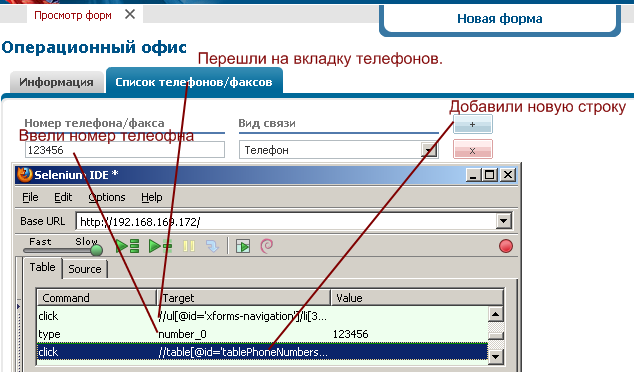
* 1. Два раза подряд нажмите ctrl+v с паузой в две секунды. Нужно нажать именно два раза, т.к. специфический плагин не хочет с первого раза принимать вставляемую дату.
  2. Убедитесь, что ваше действие записано:



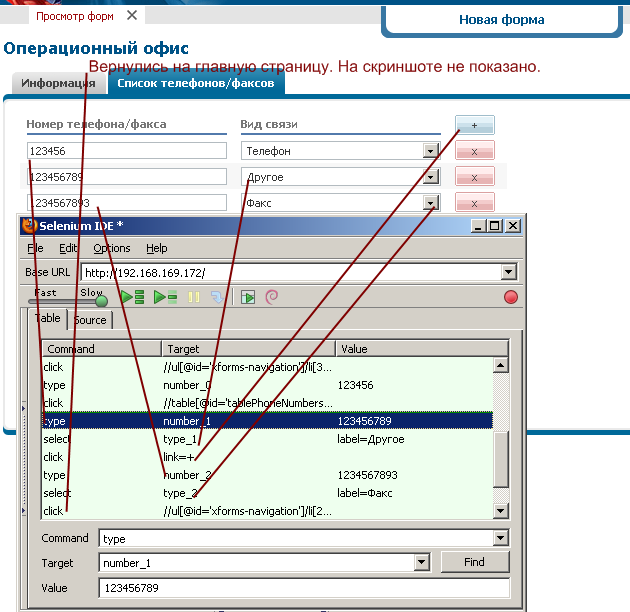
6. Перейдите на другую закладку.



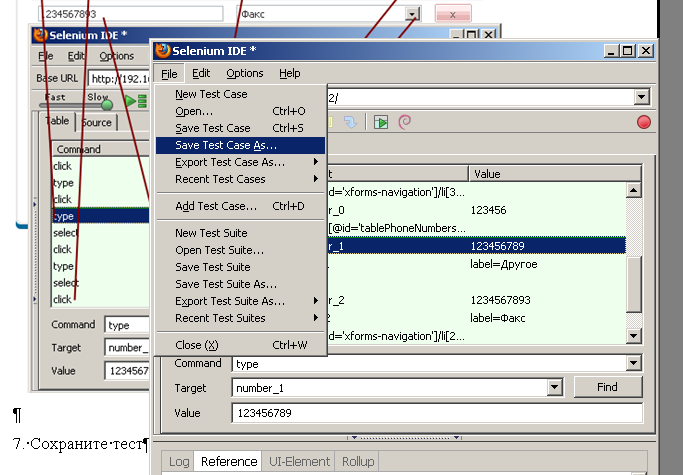
6.1. Сначала заполняйте одну строку, затем нажимайте на 



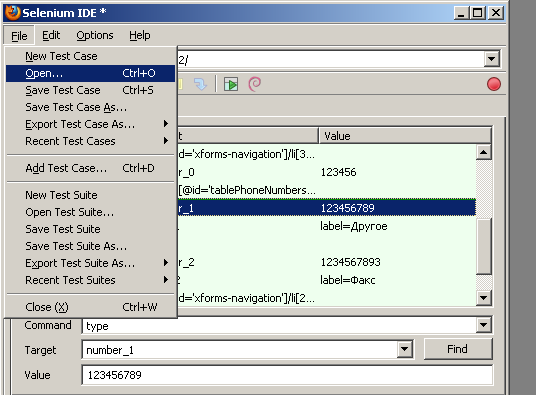
6.2. Последовательно заполните столько строк, сколько нужно.



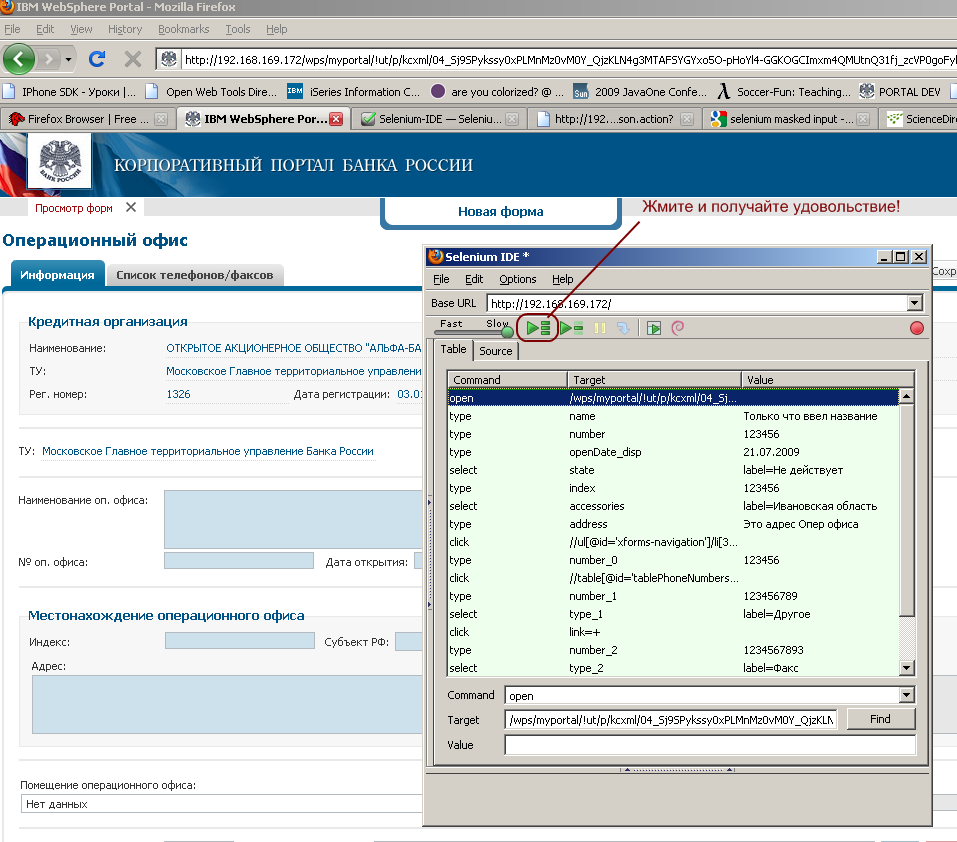
7. Сохраните тест



1. «Прогоните записанный тест»
   1. Если тест не открыт, откройте его

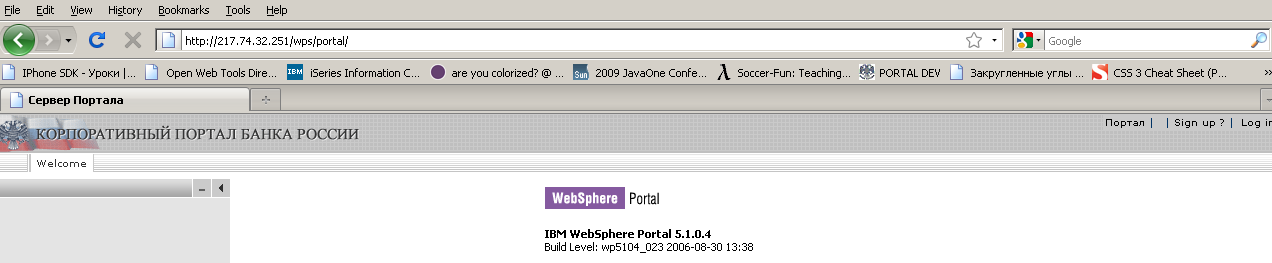


* 1. Запустите открытый тест, предварительно перейдя на нужную страницу портала

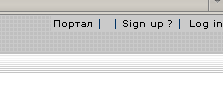


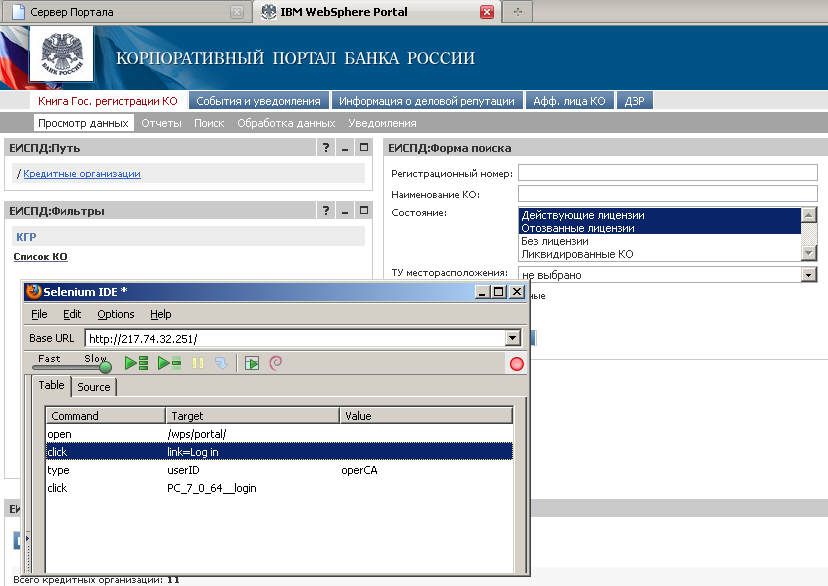
9. Как правильно начинать тест

* 1. Если вы залогинены на портале, разлогиньтесь.
  2. Урл в браузере должен быть таким: http://217.74.32.251/wps/portal/

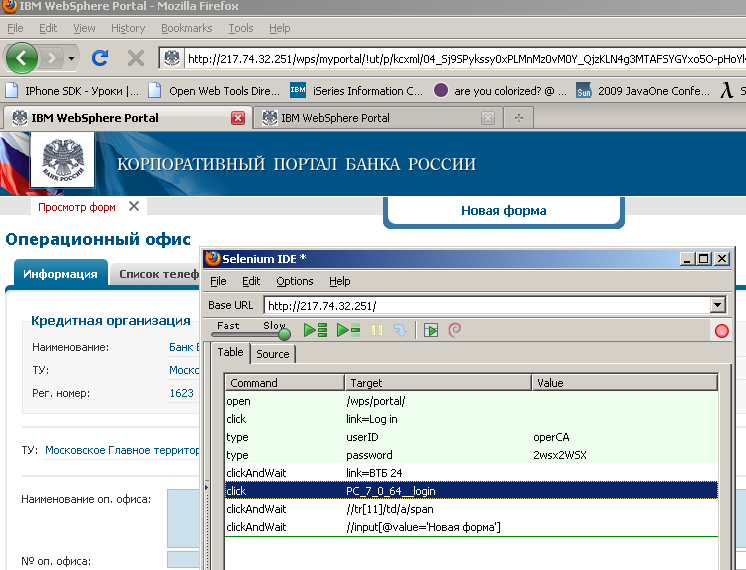


Цифры могут быть другими, но оканчиваться должен на /wps/portal/

* 1. Включите Selenium
  2. Нажмите на кнопку «запись» , она должна стать такой: 
  3. Нажмите на ссылку, чтобы попасть на страницу логина 
  4. Вводите логин и пароль, авторизуйтесь.



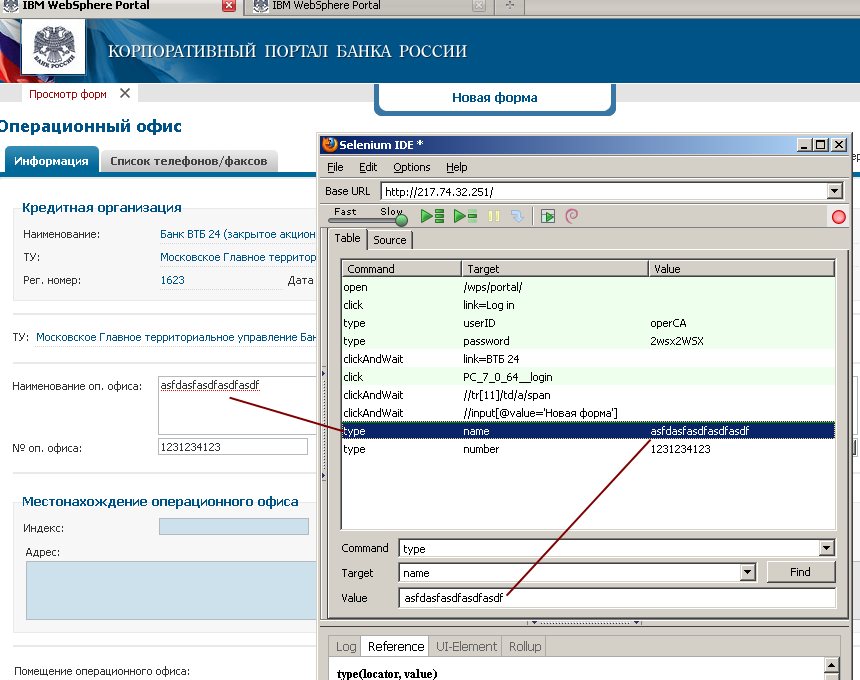
* 1. . Переходите к нужной форме и начинайте ввод данных.



9.8. Не забудьте сохранить тест (см. пункт №7)

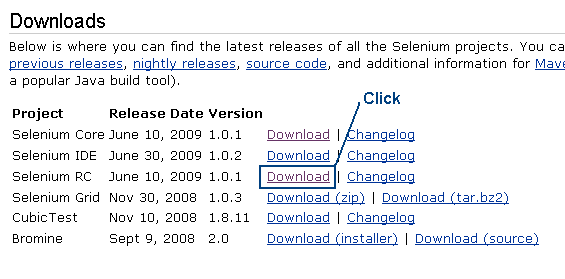
Выполнив указанные действия, сохранив тест и приложив его к багу в Джире, вы позволите программисту полностью воспроизвести сценарий ввода данных.

Вы можете модифицировать данные теста, изменяя значение поля “Value”.

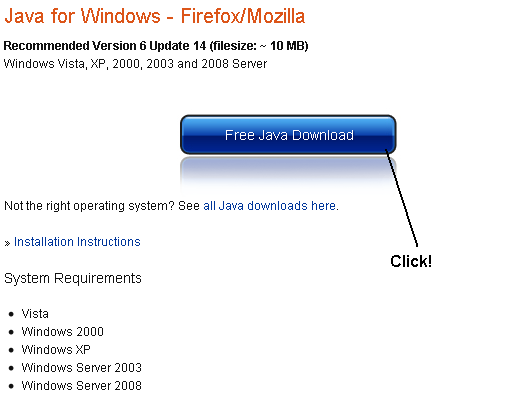


# Подготовка и запуск записанного теста под браузером Internet Explorer

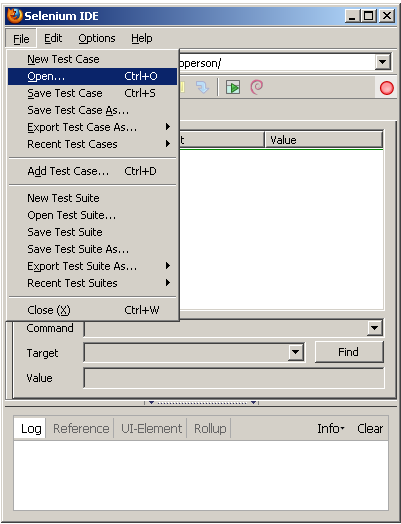
1. Установите [Selenium RC](http://seleniumhq.org/download/)



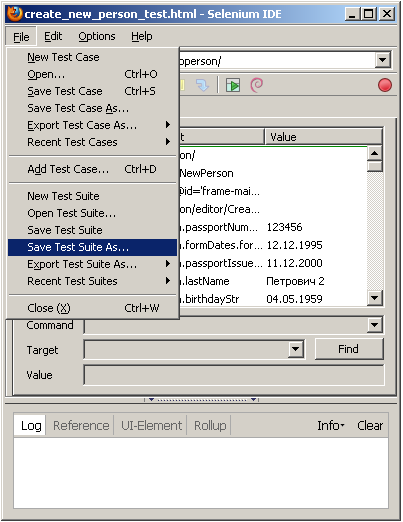
1. Скаченный архив разархивируйте, например на диск C:\ в папку selenium
2. Если у вас не установлена свежая JRE, установите, взяв [отсюда](http://java.com/en/download/inc/windows_upgrade_xpi.jsp).



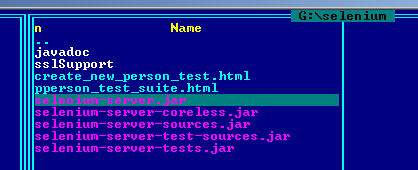
1. Откройте Far, если его нет, возьмите [отсюда](http://www.farmanager.com/download.php?l=ru).
2. Если вы не создали создали TestSuite, создайте её.
   1. Откройте ранее сохраненный тест (TestCase)



5.2. Сохраните TestSuite из одного единственного теста (TestCase). Если вы создаете тест, вы можете сразу его сохранять, как TestSuite. В одну TestSuite можно поместить несколько тестов.



5.2. откройте Far и перейдите в каталог, где разархивирован Selenium RC. На примере с картинки он установлен по пути G:\selenium



* 1. Выполните команду:

java -jar selenium-server.jar -htmlSuite "\*chrome" "http://217.74.32.251/" "G:\selenium\pperson\_test\_suite.html" "G:\selenium\result.html"

где,

\*chrome – ключ, определяющий браузер, этот ключ запускает Firefox на движке Chrome.

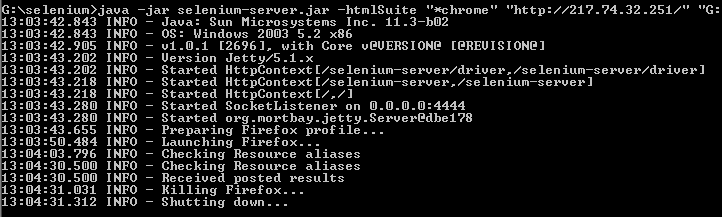
<http://217.74.32.251/> - базовый урл,

G:\selenium\pperson\_test\_suite.html полный путь к TestSuite. Рядом с сюитой должны находиться тесты, которые в неё входят.

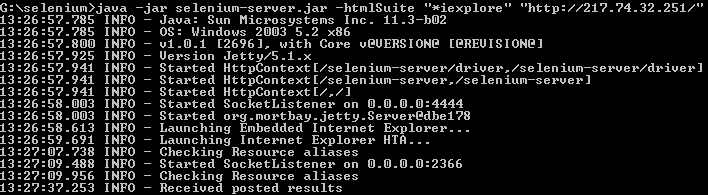
G:\selenium\result.html путь к файлу, который будет создан Селениумом. В файле будет сохранен отчет.

**Для запуска теста в браузере Internet Explorer, используйте ключ** \*iexplore

Так выглядит консоль Far’a при запуске и последующем завершении тест сюиты под браузером firefox



А так – под Internet Explorer



* 1. Если вы используете Multiple IE, т.е. у вас одновременно установлены IE6 и IE7, для запуска тестов используйте следующую команду:

java -jar selenium-server.jar -htmlSuite "\*piiexplore D:/Program Files/MultipleIEs/IE6/iexplore.exe" "http://217.74.32.251/" "G:\selenium\pperson\_test\_suite.html" "G:\selenium\result.html"

где, \*piiexplore – спец. режим для запуска IE6,

D:/Program Files/MultipleIEs/IE6/iexplore.exe – путь до exe файла с IE6 (из Multiple IE). Обратите внимание, что все слеши (/) «смотрят вправо».

Приложение 3. Использование Rational Robot

# Установка Rational Robot (RR)

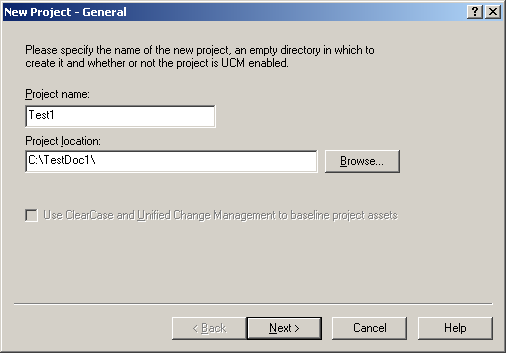
# Установка и лицензия

* + 1. В любом порядке устанавливаем продукты IBM Rational Robot, включая сам Rational Robot, Test Manager, Adminisrator, License Key Adminisrator.
    2. В конце установки IBM Rational License Key Adminisrator, необходимо выбрать, в каком виде предоставить лицензию. Чаще всего, это лиценщия в файле license.txt – нужно указать этот файл.

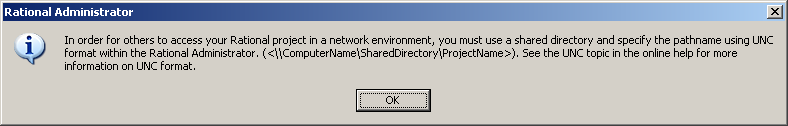
# Создание проекта в Rational Administrator

# Создание проекта

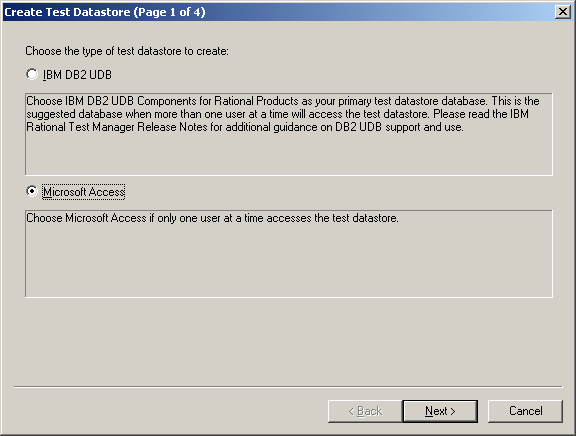
* + 1. После того, как RR полностью установлен, можно (и нужно) создать проект. Для этого создаем ПУСТУЮ папку, так, чтобы путь к этой папке и её название содержали только буквы латинского алфавита, цифры и знак \_. (При запуске скриптов по расписанию – возникают ошибки, если присутствуют русские буквы).
    2. Запустить IBM Rational Administrator.
    3. Правой кнопкой мыши нажать на слово “Projects”, в контекстном меню выбрать “ New Project”.
    4. Заполнить поле Project Name (не используя русские буквы).



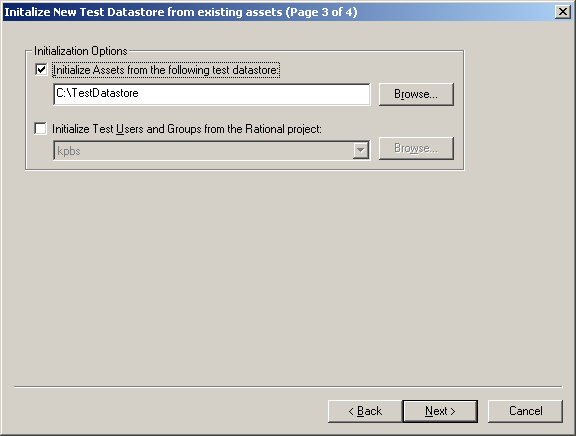
* + 1. Для заполнения поля Project Location, нажать “Browse”, выбрать папку, созданную в п.2.1.1, нажать “OK”, “Next” (если появится предупреждающее сообщение от RR, нажать “ОК” и продолжить создание проекта –оно напоминает, что для использования этого проекта с других компьютеров, необходимо расшарить папку).



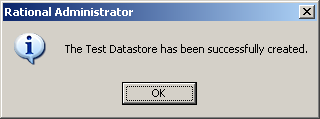
* + 1. Указать пароль для администратора проекта и подтвердить его, нажать на кнопку “Next”, “Finish”.
    2. Создание или выбор Test Datastore.
       1. Нажимаем Create, выбираем “MS Access”, нажимаем “Next”, еще раз “ Next ”.



* + - 1. Можно при создании инициализировать Test Datastore из существующей Test Datastore (иногда по-другому Test Datastore не создается). Для этого нужно выбрать “Initialize Assets from…“, инициализацию групп и пользователей “Initialize Test Users and Groups…“ можно не выбирать.



* + - 1. Нажать “Next”, “Finish”. Потом нажать “Ok”, “Ok”. Т.е. в результате должно появится сообщение, что Test Datastore успешно создана. В случае, если Test Datastore не создается, то следует попробовать инициализировать её при создании из уже существующей Test Datastore (п.2.1.7.2).



* + - 1. Можно не создавать, а выбрать Test Datastore, но она не должна быть прикреплена к проекту.

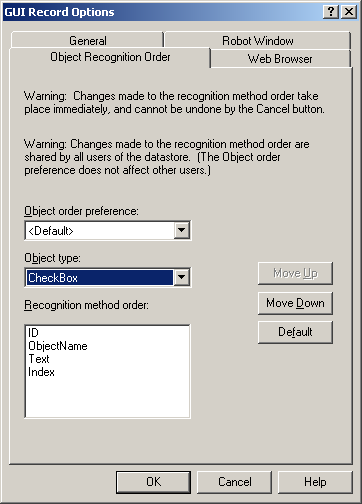
# Создание пользователей, изменение паролей

* + 1. Для создания пользователей/групп:
       1. Нажать правой кнопкой мыши на название проекта, в контекстном меню выбрать “Connect”, ввести пароль и нажать “OK”.
       2. Под названием проекта отобразится “Rational Test Datastore”
       3. Раскрыть Rational Test Datastore - нажать на “+” или дважды кликнуть мышью.
       4. Для создания нового пользователя/группы правой кнопкой мыши нажать на Test Users или Test Groups соответственно, выбрать Create… заполнить обязательные параметры и сохранить.
    2. Для редактирования пароля пользователя или включения/исключения его из группы, необходимо правой кнопкой мыши нажать на него и в контекстном меню выбрать пункт “Properties”.

# Записываем скрипты Rational Robot

# Автоматическая запись скрипта

* + 1. Настройки для более эффективной записи скрипта:
* В Rational Robot нужно выполнить следующую команду: Tools-> GUI Record Options-> Object Recognition Order. Далее для всех типов объектов последовательно выбрать тип объекта, в поле Recognition method order выбрать ID и кнопкой “Move up” поднять ID на верхнюю строчку. Это означает, что RR, при автоматическом формировании скрипта, объект данного типа будет распознаваться сначала по ID, а если RR не сможет его так распознать, то по остальным свойствам в порядке убывания.



* + 1. Создание нового скрипта:
* Запустить Rational Robot, выполнив команду File->New->Script, указать название скрипта, нажать на кнопку “OK”. Далее открывается файл для редактирования.
* Нажать на кнопку “GUI”, указать название скрипта, в автоматическом режиме провести запись действий скрипта, нажать на кнопку “Stop”. В основном окне появится этот скрипт с записанными действиями.
  + 1. Редактирование скрипта:
* Запустить Rational Robot, выполнив команду File->Open->Script, указать название скрипта, нажать на кнопку “OK”. Далее открывается файл для редактирования.
  + 1. Запись скрипта:
* Нажать на кнопку “GUI”, указать название скрипта, в автоматическом режиме провести запись действий скрипта, нажать на кнопку “Stop”. В основном окне появится этот скрипт с записанными действиями.

# Формирование скрипта вручную

* + 1. Добавление комментариев:
* Комментарием в RR является строка или часть её, начиная от первой одинарной кавычки и до конца строки.
* Если нужно закомментировать большой кусок кода, то для этого нужно выделить его и нажать Ctrl+m.
* Если нужно раскомментировать большой кусок кода, то для этого его нужно выделить и нажать Ctrl+u.
  + 1. Формирование скрипта:
* Записывать команды для RR можно вручную, после того, как скрипт открыли на редактирование. Удобно сначала все или почти все действия записать в режиме автоматического формирования скрипта, а затем отредактировать полученный код, оптимизировав, добавив верификационные точки, циклы, условия и, при необходимости, подключив библиотеки.
  + 1. Команды для работы с IE:
* Запуск любого приложения StartApplication. Запуск IE и переход, на пример, адресу www.mail.ru:

**StartApplication "C:\Program Files\Internet Explorer\IEXPLORE.EXE** [**http://www.mail.ru**](http://www.mail.ru/)**"**

* Для случая, если перед следующим действием, необходимо сначала подождать когда загрузится или обновится страница, подходит команда Browser NewPage:

**Browser NewPage,"HTMLTitle=Сервер портала;Index=0",""**

* DelayFor – команда, позволяющая остановит выполнение скрипта на указанное, в качестве параметра, время в миллисекундах. Пример, остановка на 3 секунды:

**Delayfor 3000**

* + 1. Verification Point (VP) и их альтернатива.
* Скрипт, записанный без проверок “правильности” работы приложения/программы является бесполезным, много времени уйдет на то, чтобы понять, почему скрипт не отработал – ошибка в скрипте или программе. Для отслеживания поведения программы применяются VP. Добавить их можно, если выполнить команду в RR: Insert->Verification Point-> Тип VP. Наиболее часто используются Alphanumeric, Object Data, Object Properties. Не следует использовать Window Image, Region Image из-за их нестабильной работы и привязанности к координатам. Подробнее о VP можно прочитать в Help (SQABasic Reference) Rational Robot.
* У VP есть ряд значительных недостатков, таких как: их нельзя использовать в библиотеках, иногда невозможно управлять параметрами VP, а так же сложно использовать в различных циклах. Поэтому удобнее использовать следующий набор команд:

**If (Условие) Then**

**SQALogMessage sqaPass, "Сообщение в логах – ошибки нет", "Проверка комментариев"**

**else**

**SQALogMessage sqaFail, " Сообщение в логах – ошибка есть ", " Проверка комментариев "**

**end If**

где SQALogMessage – записывает в логи сообщение, подсвеченное:

sqaPass - зеленым

sqaFail – красным (цвет ошибки)

sqaNone – белым

* + 1. Циклы.
* While (Условия) Действия Wend, пример:

**While j<>9**

**j=j+1**

**m=m+k**

**ListBox Click, "Name=licensestate", "Text=Отзыв"**

**ListBox Ctrl\_Click, "Name=licensestate", "Text=Действует"**

**ListBox Ctrl\_Click, "Name=licensestate", "Text=Приостановлена"**

**Wend**

* If (Условия) Действия Else Действия End if, пример:

**If j<>9**

**j=j+1**

**m=m+k**

**ListBox Click, "Name=licensestate", "Text=Отзыв"**

**Else**

**ListBox Ctrl\_Click, "Name=licensestate", "Text=Действует"**

**ListBox Ctrl\_Click, "Name=licensestate", "Text=Приостановлена"**

**End if**

* For i=M to N step K Действия Next i, где M, N, K задаются в скрипте, пример:

**For i=2 to 10 step 1**

**ListBox Click, "Name=licensestate", "Text=Отзыв"**

**ListBox Ctrl\_Click, "Name=licensestate", "Text=Действует"**

**ListBox Ctrl\_Click, "Name=licensestate", "Text=Приостановлена"**

**Next i**

* + 1. Библиотеки.
* Библиотека состоит из двух файлов:
  + Файл заголовка (с расширением sbh), где содержатся объявления процедур/функций и информация о библиотеке в которой она содержится.
  + Файл библиотеки, где содержится исходный код процедур/ функций (с расширением sbl).
* Могут присутствовать и динамические файлы библиотек связи dll.
* Файлы создаются командой File->New->SQABasic file, далее выбирается вида файла.
* Подключение библиотек осуществляется директивой ‘include в скрипте или другой библиотеке.
* В библиотеках не могут присутствовать VP.
* Имена процедур, функций, переменных, констант и меток должны быть не длиннее 40 символов, начинаться с букв латинского алфавита, могут содержать цифры и знак \_

Пример header-файла:

**Declare Sub Copy\_simple BasicLib "GoToExcelNew" (ExcelCoord as String,ExcelFile as String)**

Пример library-файла:

**Sub Copy1\_simple (coord1 as string, coord2 as string)**

**Dim Result As Integer**

**Dim coord As String**

**delayfor 100**

**Window SetContext, "Caption=Microsoft Excel - "&coord2, ""**

**delayfor 100**

**InputKeys coord1**

**End Sub**

Подключение библиотеки в скрипте и вызов процедуры:

**'$Include: "GoToExcelNew.sbh"**

**Sub Main**

**Dim Result As Integer**

**...**

**call Copy1\_simple (StrCoord, sTextFile )**

**...**

**End Sub**

# Запускаем и отлаживаем скрипты

# Отладка скрипта

При отладке скрипта удобно использовать следующие команды, находящиеся в меню Debug RR:

* Go – запускает активный в данный момент скрипт
* Go Until Cursor - запускает активный в данный момент скрипт, с начала и до строки, на которой находится курсор
* Animate –– запускает активный в данный момент скрипт , указывая желтой стрелкой в скрипте, какая строка сейчас выполняется
* Pause – приостанавливает выполнение скрипта.
* Stop – останавливает выполнение скрипта
* Set or Clear Breakpoint – используется для добавления/удаления точки останова. При добавлении в скрипт точки останова, на строке, где она установлена, появляется красная точка слева. При удалении – точка пропадает.
* Clear All Breakpoints – удаляются все точки останова из активного в данный момент скрипта.
* и т.д.

# Выполнение из RR

* + 1. Настройки RR

Выполнить команду Tools->GUI Playback options. Наиболее важными являются следующие настройки:

* Playback->delay between commands, delay between keystrokes – это “скорость” работы RR во время выполнения скрипта, в процессе отладке лучше устанавливать интервалы побольше – порядка 200 – 500 миллисекунд. Дальнейшие настройки зависят от тестируемой системы
* Wait State –> Timeout after - время ожидания “правильного” ответа системы на действия RR. Если в системе возможны сбои, но скрипт должен выполняться, то лучше сделать Timeout after побольше, например, 300-500 секунд.
  + 1. Выполнение скрипта

Запустить скрипт можно, если выполнить команду: File->Playback (или нажать на кнопку Playback в виде треугольника в верхнем меню), далее выбрать скрипт, нажать “OK”, уточнить информацию по логам (либо писть их поверх старых, либо – указать новое название, папку), нажать “OK”.

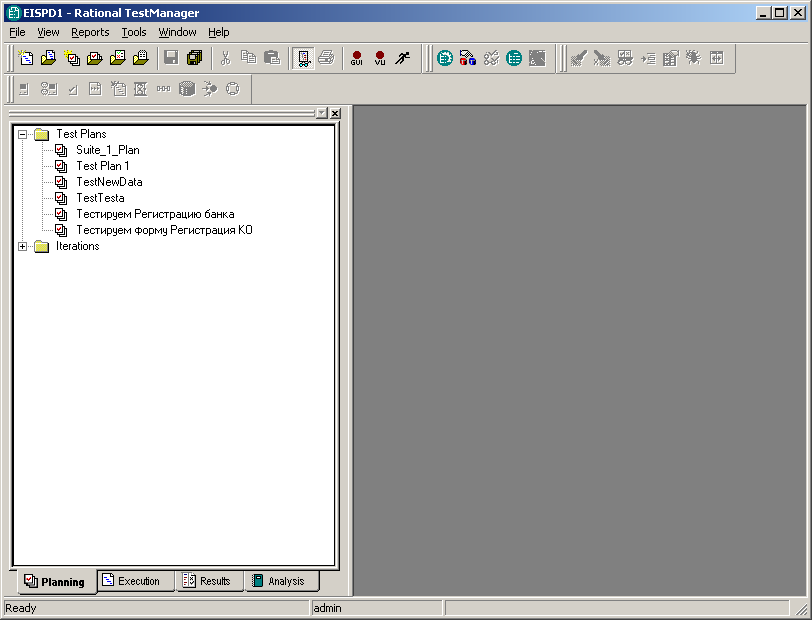
Либо через запуск Suite, см. п.5

После окончания выполнения скрипта, открывается Test Manager с логами, которые можно сохранять, для ведения истории тестирования или же удалять.

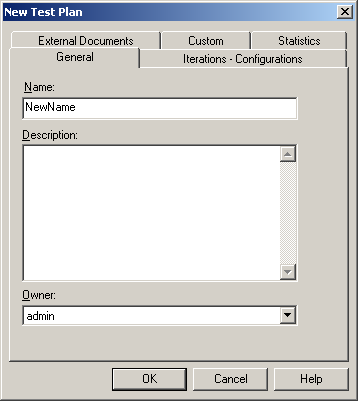
# Запускаем скрипты из TestManager

# Создание TestPlan

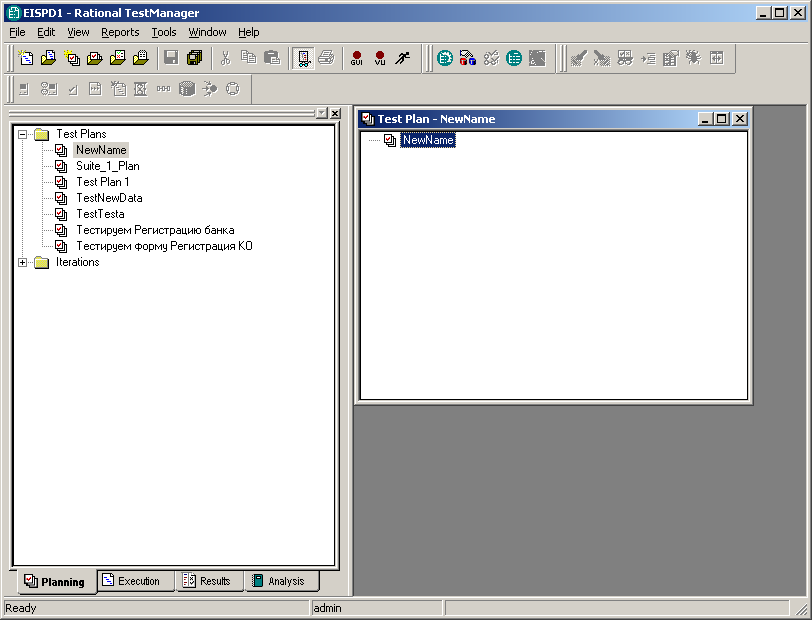
* + 1. Запускаем TestManager.
    2. Переходим на вкладку Planning в левом нижнем углу.



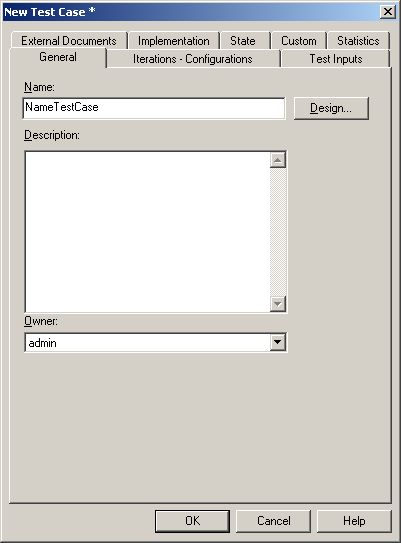
* + 1. Правой кнопкой мыши нажимаем на надпись “Test Plans”.
    2. В контекстном меню выбираем “New Test Plan”.
    3. Обязательно вводим его название (на английском языке), при необходимости – заполняем остальные параметры, нажимаем на кнопку “ОК”.

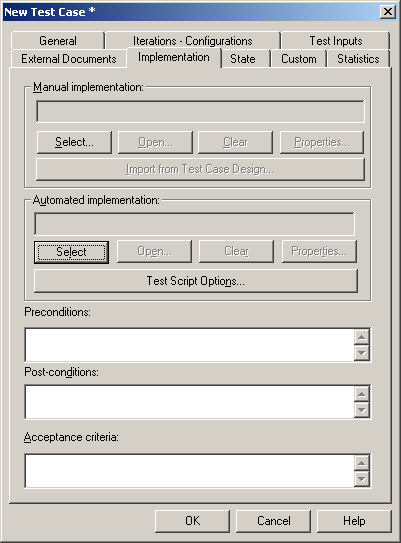


* + 1. Тест-план с сохраненным названием появляется в списке “Test Plans”.
    2. Нажимаем на название левой кнопкой мыши дважды (открываем тест-план).
    3. В правой части окна Test Manager появляется окно “Test Plan – Название тест-плана”.



* + 1. Нажимаем правой кнопкой мыши на папку Default (это Test Case Folder).
    2. Если её в списке нет, то нажимаем правой кнопкой мыши на название тест-плана и выбираем “Insert Test Case Folder”, обязательным будет поле название.
    3. В контекстном меню выбираем либо создание вложенной папки (для логического разделения Test Case) либо сразу создаем Test Case.
    4. В случае, если создаем папку, то обязательным будет поле название.
    5. Если создаем Test Case, то необходимо заполнить Название (в закладке General), а так же выбрать скрипт (закладка Implementation, в поле Automated Implementation нажать на кнопку “Select”.

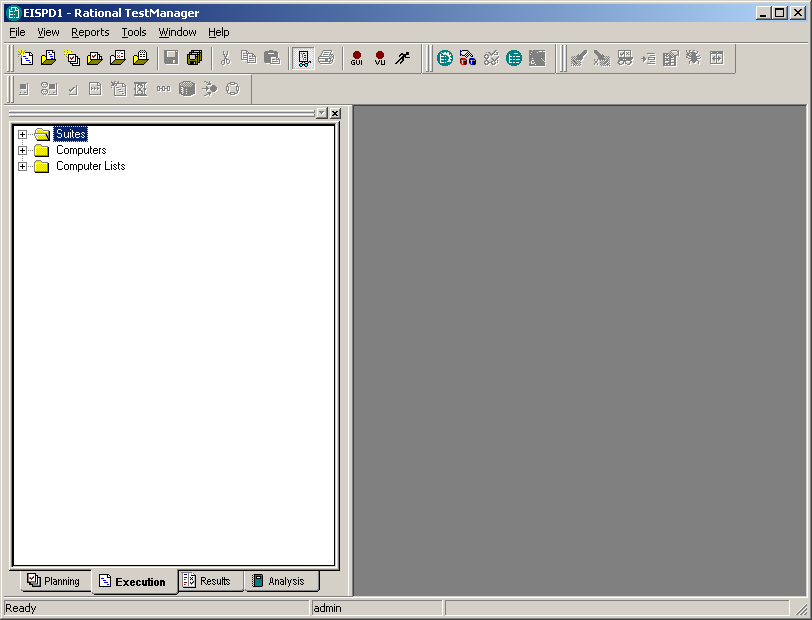




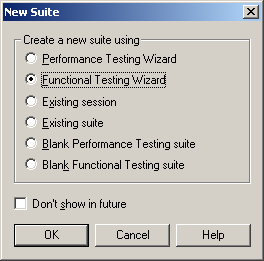
* + 1. Выбрать тип скрипта “GUI”, выбрать нужный скрипт, нажать на кнопку “OK”,еще раз “OK”.
    2. После этого сохраненный test case появится в выбранной ранее папке.
    3. Та же процедура повторяется для всех скриптов, которые будут запускаться.

# Создание Suite

* + 1. Переходим на вкладку Execution в левом нижнем углу.



* + 1. Нажимаем правой кнопкой мыши на папку “Suites”.
    2. В контекстном меню выбираем New Suite.
    3. В появившимся окне выбираем “Functional Testing Wizard”.



* + 1. Нажимаем “OK”.
    2. Для выбора необходимых для запуска Test Case, нажимаем “Select”.
    3. Выбираем скрипты (можно сразу все), нажимаем “ОК”.
    4. Нажимаем “Next”, еще раз “Next”, “Finish”.
    5. Теперь сохраняем появившийся Suite (Необходимо указать его название).
    6. Он появляется в списке Suites.

# Запуск Suite

* + 1. Если нажать на название правой кнопкой мыши, то в контекстном меню можно выбрать “Run” – это запуск скрипта и всех включенных в него скриптов (test case).

# Редактирование Suite

* + 1. Открываем Suite – дважды нажав на него мышкой или выбрав в контекстном меню – Open.
    2. При необходимости – удаляем из него Test Case.
    3. Допустим, необходимо изменить порядок выполнения тест-кейсов. Чтобы после TestCase1 шел TestCase2. Для этого правой кнопкой мыши нажимаем на TestCase1, в контекстном меню выбираем “Insert Test Case”, выбираем нужный тест-кейс и нажимаем “OK”.

# Запускаем по расписанию

Чтобы запустить Suite по расписанию, необходимо создать файл с расширением .bat, и следующим содержанием, далее следует пример:

"C:\Program Files\Rational\Rational Test\rtmanager.exe" "SuiteName" /runsuite /user admin /password 1 /project C:\temp\proj1.RSP /COMPUTERS LOCAL /BUILD "BUILD 1" /LOGFOLDER NadinTest /LOG ResultSuiteName1 /overwritelog /CLOSE

Где:

1. "C:\Program Files\Rational\Rational Test\rtmanager.exe”

путь до файла rtmanager.exe, заключается в “”, rtmanager.exe должен быть размером около 3 Мб (если значительно меньше – это ошибка, необходимо либо скопировать этот файл с другой машины или переустановить RR)

1. Далее следует пробел
2. "SuiteName" – имя запускаемого Suite
3. Далее следует пробел
4. /runsuite - это команда
5. Далее следует пробел
6. /user admin – вместо admin можно указать имя другого пользователя проекта, из-под которого будет запускаться Test Manager
7. Далее следует пробел
8. /password 1 – вместо 1 указывается пароль пользователя для входа в систему
9. Далее следует пробел
10. /project C:\temp\proj1.RSP – путь до файла проекта с расширением .rsp
11. Далее следует пробел
12. /COMPUTERS LOCAL – указываем, где будет запускаться suite
13. Далее следует пробел
14. BUILD "BUILD 1" – как называется build, в котором будут логи
15. Далее следует пробел
16. /LOGFOLDER NadinTest - папка, в которой будут логи
17. Далее следует пробел
18. /LOG ResultSuiteName1 - название логов
19. Далее следует пробел
20. /overwritelog /CLOSE – после повторного запуска системы, логии пишутся вместо стрых, после выполнения скрипта, RR закрывается

ВНИМАНИЕ: в адресах не должно быть названий папок с русскими буквами!

Теперь создаем задачу в Scheduled Tasks:

Пуск->Settings->Control panel-> Scheduled Tasks-> Add Scheduled Tasks далее необходимо добавить в задачи файл .bat и указать время и период выполнения задачи