Реферат

Это пример каркаса расчётно-пояснительной записки, желательный к использованию в РПЗ проекта по курсу РСОИ.

Текст в документе носит совершенно абстрактный характер.

Содержание

Глоссарий

Распределённый — Слово, которое нельзя употреблять. Но надо протестировать длинные строки в глоссарии.

Обозначения и сокращения

АИС — Автоматизированная информационная система.

Введение

Данный дипломный проект ориентирован на практическое применение навыков автоматизации тестирования: обзор и анализ существующих методов и средств нагрузочного тестирования, разработку программного обеспечения, в значительной степени снижающего сложность осуществления функционального и нагрузочного тестирования взаимодейтсвия клиента и сервера по AMF-протоколу, разработку методики функционального и нагрузочного тестирования с использованием предложенного ПО.

В настоящее время активно развивается рынок Web-приложений. Все больше различных услуг предоставляется через Интернет, классические информационные системы в различных компаниях также становятся Web-приложениями, предоставляя для своих сотрудников и партнеров доступ к ресурсам компании не только из локальной, но и из глобальной сети. Концепции и технологии, используемые при разработке web-приложений, постоянно развиваются и совершенствуются. Оптимизируется использование ресурсов и времени, улучшаются возможности по отображению предоставляемой информации, динамичность и интерактивность web-приложений.

На сегодняшний день одним из наиболее перспективных подходов к обеспечению всего вышеперечисленного является концепция Rich Internet Application (в дальнейшем RIA) - это приложения, доступные через сеть Интернет, обладающие особенностями и функциональностью традиционных настольных приложений.

Приложения RIA, как правило:

а) передают веб-клиенту необходимую часть пользовательского интерфейса, оставляя большую часть данных (ресурсы программы, данные и пр.) на сервере;

б) запускаются в браузере и не требуют дополнительной установки программного обеспечения;

в) запускаются локально в среде безопасности, называемой «песочница» (sandbox).

На Рис. 0.1 демонстрируется место Rich Internet Applications среди других технологий программных систем

Рисунок 0.1 — RIA среди других технологий

Термин «RIA» впервые был упомянут компанией Macromedia в официальном сообщении в марте 2002 года. Подобные концепции существовали и несколькими годами ранее, например:

а) DHTML — HTML страницы с динамическим содержимым;

б) Java-applet — это программы написанные на языке Java, как правило, предназначенные для загрузки посредством браузера;

Для начала рассмотрим архитектурные особенности построения web-приложений базирующихся на RIA концепции. Для этого сравним принцип работы традиционного web-приложения и RIA-приложения (Рис. 0.2).

Работа традиционных web-приложений сконцентрирована вокруг клиент серверной архитектуры с тонким клиентом. Такой клиент переносит все задачи по обработке информации на сервер, а сам используется в основном для отображения статического контента. Основной недостаток этого подхода в том, что все взаимодействие с приложением должно обрабатываться сервером, что требует постоянной отправки данных на сервер, ожидания ответа сервера, и загрузки страницы обратно в браузер. В отличие от традиционного web-приложения в RIA значительная часть функционала исполняется на стороне клиента, поэтому появляется возможность отправлять и получать данные с сервера только по мере

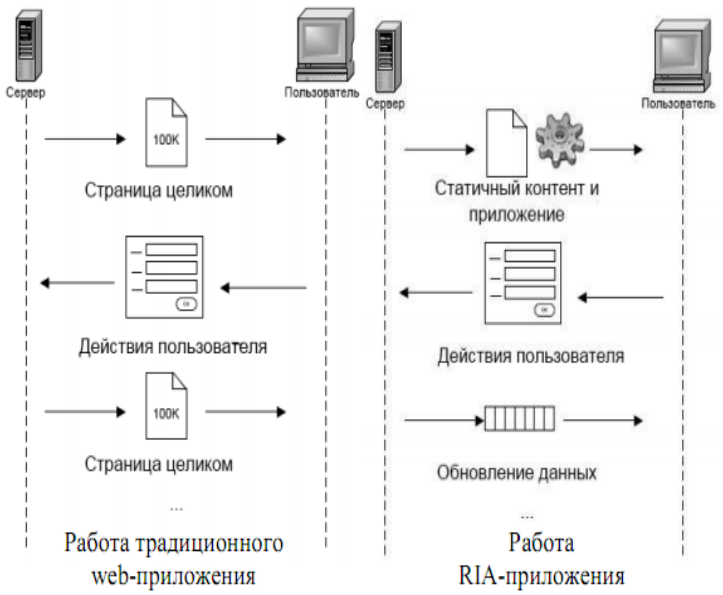

Рисунок 0.2 — Принципы работы традиционного web-приложения и RIA приложения

необходимости. Если подключение нестабильно, клиентская часть RIA-приложения обладает возможностями кэширования данных и работы без подключения к сети в режиме offline.

На основе вышесказанного, можно выделить ряд преимуществ RIA как перед настольными приложения, так и перед стандартными веб-приложениями:

а) для работы с RIA не требуется установка приложения, как правило, все что необходимо, это установка плагина для браузера;

б) пользователи могут использовать приложение на любом компьютере, имеющем соединение с Интернет, причем обычно не важно, какая операционная система на нем установлена;

в) интерфейс RIA обладает большей интерактивностью, он не ограничен лишь использованием языка HTML, применяемого в стандартных веб-приложениях. Наиболее сложные приложения RIA предлагают внешний вид и функциональность, близкие к настольным приложениям;

г) при работе веб-приложения компьютер пользователя гораздо меньше подвержен риску вирусному заражению, чем при запуске исполняемых бинарных файлов;

д) использование вычислительных ресурсов клиента и сервера лучше сбалансировано;

е) взаимодействие клиента с сервером осуществляется без ожидания каких-либо действий пользователя.

Для реализации RIA многими IT-компаниями предлагаются различные технологии. Наиболее известными из них являются AJAX, Flash, Flex (в дальнейшем Flex) и AIR фирмы Adobe; ActiveX, WPF и Silverlight корпорации Microsoft; Java FX и Java Applets компании Sun Microsystems .

В настоящее время одной из наиболее распространенных технологий разработки RIA является Adobe Flex, которая представляет собой среду с открытым кодом для создания и обслуживания web-приложений, совместимую со всеми распространенными браузерами, платформами персональных компьютеров и версиями операционных систем и имеющую максимальную поддержку со стороны разработчиков.

Flex представляет собой набор классов, расширяющих возможности Flash. Flex позволяет описывать интерфейс web-приложения на MXML – декларативном языке описания и настройки свойств визуальных элементов интерфейса, основанном на XML. Логика web-приложения пишется на ActionScript – полноценном объектно-ориентированном языке программирования, который является одним из диалектов ECMAScript. Результатом компиляции является файл SWF, предназначенный для выполнения в браузере (на платформе Flash Player, которая существует в виде плагина к браузеру) или как самостоятельное приложение (на платформе AIR).

Flex-framework включает возможности локализации, стилизации приложения, разработки модульного приложения, встроенные валидаторы и форматоры текстовых полей — все те инструменты, которые нужны разработчикам приложений, работающих online. Также Flex предоставляет полные мультимедийные возможности Flash Platform, такие как потоковое мультимедиа, возможность получить доступ к веб-камере и микрофону пользователя, бинарные сокеты, обширные возможности сетевых коммуникаций (HTTP-запросы, веб-сервисы, встроенный формат

сериализации AMF), оперирование координатами трехмерного пространства).

Вся разработка во Flex ориентирована на применение готового набора расширяемых компонентов, внешний вид которых позволяет гибко настраивать CSS, что облегчает задачу разработчика.

Flex SDK является бесплатным инструментарием с июня 2007 года с открытым исходным кодом, распространяемым на условиях Mozilla Public License. Для работы с процедурами и классами этого фреймворка можно использовать как бесплатные (Eclipse WTP IDE, FlashDevelop IDE) так и платные (Flex Builder IDE, IntelliJ IDEA IDE, Aptana Studio IDE, PowerFlasher FDT IDE) среды разработки.

Flex приложения предоставляют возможность реализации клиент-серверного взаимодействия на основе бинарного формата обмена данными — AMF(Action Message Format). AMF используется для сериализации структурированных данных, таких как объекты Action Script или XML, и обмена сообщениями между Adobe Flash клиентом и удалённым сервисом. AMF более экономичен по трафику по сравнению с XML и позволяет передавать типизированные объекты.

Как известно огромную роль в жизненном цикле программного обеспечения играет фаза тестирования. Тестирование — один из важнейших этапов контроля качества в процессе разработки ПО. Автоматизированное тестирование является его составной частью. Оно использует программные средства для выполнения тестов и проверки их результатов, что помогает сократить время тестирования и упростить его процесс, а также может дать возможность выполнять определенные тестовые задачи намного быстрее и эффективнее чем это может быть сделано вручную. Автоматизация тестирования даёт следующие преимущества:

а) Автоматизация — выполнение сложно воспроизводимых вручную тестов, таких как нагрузочное и стресс-тестирование;

б) Меньшие затраты на поддержку — когда автоматические скрипты уже написаны, на их поддержку и анализ результатов требуется, как правило, меньшее время чем на проведение того же объема тестирования вручную;

в) Повторяемость — все написанные тесты всегда будут выполняться однообразно, то есть исключен «человеческий фактор». Тестировщик не пропустит тест по неосторожности и ничего не напутает в результатах;

г) Быстрое выполнение — автоматизированному скрипту не нужно сверяться с инструкциями и документациями, это сильно экономит время выполнения;

д) Отчеты — автоматически рассылаемые и сохраняемые отчеты о результатах тестирования.

Если сложить все вышесказанное, то больший объем тестирования, может быть достигнут с меньшими усилиями, давая при этом возможность улучшать как качество, так и продуктивность.

Однако использование AMF вызывает ряд трудностей для реализации автоматизации функционального и нагрузочного тестирования взаимодействия сервера и Flex клиента, связанных с бинарной природой протокола. Так как amf сообщение представляет собой совокупность байтов, разработчикам и тестировщикам трудно считывать и изменять содержащуюся в нём информацию.

Отдельной проблемой является нагрузочное тестирование таких приложений — имитация работы с приложением большого количества пользователей за счёт запуска набора тестов в несколько потоков. Сложность состоит в поддержке уникальности идентификатора сессии каждого клиента. Идентификатор клиента выделяется строго одному подключенному SWF-файлу и прошит в передаваемых данных. Стандартный порядок тестирования — запись запросов пользователя с использованием прокси-сервера и запуск сохранённого сценария в несколько потоков - в данном случае будет неэффективным, так как все перехваченные запросы будут иметь одинаковый идентификатор клиента и тесты пройдут только в 1 потоке, а остальные будут возвращены с ошибкой о неправильной сессии.

Таким образом, можно сформулировать наличие противоречия между необходимостью проведения функционального и нагрузочного тестирования и сложностью его реализации на технлогии Flex и поставить следующие задачи на дипломную работу:

а) разработка программного обеспечения, в значительной степени снижающего сложность осуществления функционального и нагрузочного тестирования взаимодейтсвия клиента и сервера по AMF-протоколу;

б) разработка методики функционального и нагрузочного тестирования с использованием предложенного ПО.

1 Анализ существующих решений

В этом разделе выполнены следующие работы: анализ существующих решений в области автоматизации тестирования, сравнительная оценка вариантов возможных решений, выбор наиболее оптимального средства автоматизации.

1.1 Формулировка критериев выбора

На сегодняшний день существует ряд различных программных комплексов, предоставляющих возможность для проведения функционального, нагрузочного, регрессионного тестирования Flex приложений, поэтому для решения поставленной на дипломный проект задачи нет необходимсти создавать тестовую утилиту с нуля, разумнее будет выбрать одно из существующих решений, и,в случае, если оно не полностью удовлетворяет нашим условиям, доработать его. Чтобы выбрать из предложенного разнообразия подходящий продукт, сформулируем ряд требований, которым должна удовлетворять выбранная нами тестовая утилита.

а) Возможность проведения нагрузочных тестов. Так как большинство web-приложений являются многопользовательскими, проведение нагрузочных тестов является обязательным условием качественного тестирования. Поскольку запуск тестов в несколько потоков является довольно сложной задачей, инструмент для автоматизации должен обеспечивать удобный и надёжный способ многопоточного прогона тестовых сценариев.

б) Поддержка протокола AMF. Утилита должна осуществлять тестирование взаимодействия Flex приложения с сервером, предоставляя пользователю возможность наблюдать трафик. Тестирование пользовательского интерфейса для нашей задачи нецелесообразно.

в) Простота создания тестовых сценариев. Утилита должна предоставлять пользователю удобных механизм записи и редактирования тестов. такими как maven и ant, и серверами интеграционного тестирования. Это условие

г) Возможность запуска тестов из командной строки. Данная возможность позволяет выполнять тесты в автоматическом режиме в системах непрерывной интеграции, что является необходимостью, особенно в тех случаях, когда над разными частями тестируемой системы разработчики трудятся независимо и необходимо выполнении частых автоматизированных сборок проекта для скорейшего выявления и решения интеграционных проблем. Часто производители тестовых фреймворков предоставляют свои продукты для проведения интеграционного тестирования, однако зачастую они узко заточены под конкретный круг задач, поэтому нас будет интересовать возможность запуска тестов на таких инструментах интеграции, как Jenkins и Hudson, которые на данный момент широко распротранены и имеют множество плагинов, позволяющих значительно расширить их существующую функциональность.

д) Кроссплатформенность. Выбранная нами тестовая утилита должна работать под управлением различных операционных систем.

е) Условия распростронения продукта. В идеале программное обеспечение должно быть бесплатным и иметь открытый исходный код с возможность создания собственных расширений, это является одним из основных критериев отбора, так как тестировщики и разработчики всегда должны иметь возможность доработки и усовершенствования существующего функционала тестового фреймворка, чтобы адаптировать его под специфику работы тестируемого ими приложения.

ж) Документация. Так как современные инструменты тестирования не менее сложны, чем среды разработки, большим плюсом будет наличие подробной пользовательской документации, документации для разработчиков, различных форумов и сайтов, посвящённых данному инструменту тестирования.

Далее в этой главе будет проведён детальный анализ существующих тестовых утилит на основе выделенных критериев.

1.2 Обзор утилит для тестирования Flex приложений

1.2.1 HP QuickTest Professional

HP QuickTest Professional (QTP) — один из инструментов автоматизации функционального тестирования, является флагманским продуктом компании HP в своей линейке. Для разработки автоматизированных тестов QTP использует язык VBScript(Visual Basic Scripting Edition) — скриптовый язык программирования, интерпретируемый компонентом Windows Script Host. Он широко используется при создании скриптов в операционных системах семейства Microsoft Windows. QTP поддерживает ряд технологий, среди которых есть и Macromedia Flex. Поддержка Flex осуществляется засчёт установки плагина, предоставляемого компанией Adobe (Flex QTP add-in).

Чтобы приступить к тестированию приложение необходимо скомпилировать, создав для него HTML-оболочку, и развернуть его либо локально, либо на web-сервере. Создание тестов в QTP осуществляется следующим образом: приложение открывается в браузере и все действия, совершаемые пользователем с пользовательским интерфейсом приложения записываются в виде строчек Visual Basic скрипта. QTP поддерживает запись большинства наиболее часто используемых во Flex приложениях событий (операций), связанных с пользовательским интерфейсом. Однако часть из них, например некоторые атомарные операции, игнорируются во время записи тестов. Пользователь имеет возможность добавить их в текст скрипта вручную. Для проверки правильности выполнения теста пользователь задаёт ожидаемые значения для выполняемых операций - checkpoints. Во время автоматического прогона тестов запускать браузер уже не нужно, достаточно лишь указать HTML страницу, используемую для тестов. Для выявления причины возникновения ошибок в тестах или каких-либо других неполадок можно обратиться к логам Flash Player или настроить и включить логирование в QTP. Возможность прогона тестов в несколько потоков в QTP отсутствует

Для проведения тестов с HP QuickTest Professional необходимо использовать браузер Internet Explorer версии 6 или выше. HP QuickTest

Professional работает с операционными системами семейства Windows и является платным программным обеспечением.

1.2.2 IBM Rational Functional Tester

IBM Rational Functional Tester является средством автоматизированного регрессивного тестирования, предназначенным для тестирования Java, .NET, Web-приложений, включая Flex-приложения, и терминальных приложений на платформах Windows и Linux. Functional Tester поддерживает два языка сценариев: Java и Visual Basic.NET. Для тестирования Java-приложений в программу Functional Tester включена открытая среда разработки Eclipse. Установка дополнительных компонентов не требуется. Если требуется использовать язык сценариев Visual Basic.NET, перед установкой IBM Rational Functional Tester необходимо установить Visual Studio.NET.

Перед началом тестирования приложение загружается в браузере, затем все действия,совершаемые пользователем записываются в виде Java или Visual Basic.NET скрипта (в зависимости от настроек). Пользователь может редактировать скрипты, а также задавать ожидаемые результаты выполнения той или иной операции для проверки работы приложения. Как и QTP, IBM Rational Functional Tester во время записи тестов добавляет компоненты приложения в свой набор объектов. Далее тесты можно запускать в автоматическом режиме. Также в IBM Rational Functional Tester имеется возможность многократного запуска тестов с различным набором данных. Если стандартного набора объектов(таких как кнопки, текстовые поля и т.д.) недостаточно, существует возможность самостоятельно добавить во фреймворк необходимые объекты. Это возможно благодаря Rational Functional Tester proxy software development kit (SDK), который имеет удобное API и довольно подробную сопроводительную документацию. Также IBM Rational Functional Tester упрощает механизм регрессионного тестирования. Functional Tester использует усовершенствованную технологию ScriptAssure для того, чтобы «изучить» контрольные характеристики пользовательского интерфейса, что позволяет идентифицировать те же

самые средства управления в новой версии, несмотря на внесенные изменения. Эти характеристики сохраняются в объектной карте, совместный доступ к которой могут получить различные скрипты и участники проекта. Благодаря этой карте изменения, внесенные в характеристики распознавания объекта, будут отражены во всех скриптах тестирования, что существенно упрощает обслуживание.

Rational Functional Tester поддерживает операционные системы Windows 2000, XP, Vista, 7, Linux. Является платным программным обеспечением.

1.2.3 NeoLoad

NeoLoad-профессиональный инструмент нагрузочного тестирования веб-приложений, в том числе и Flex, работающий на многих платформах, в том числе Windows, Solaris, Linux. Осуществляя моделирование большого числа пользователей, которые обращаются к приложению, NeoLoad делает проверку надежности и производительности приложения при разных нагрузках.

Тестирование Flex приложений осуществляется засчёт записи AMF трафика во время взаимодействия пользователя с приложением. Все перехваченные запросы отображаются в xml формате, что позволяет редактировать содержащиеся в них данные. Далее записанные тесты могут быть запущены в несколько потоков.

NeoLoad является платным программным обеспечением с закрытым исходным кодом.

1.2.4 Apache JMeter

JMeter - инструмент для проведения нагрузочного тестирования, изначально разрабатывался как средство тестирования web-приложений, в настоящее время он способен проводить нагрузочные тесты для JDBC-соединений, FTP, LDAP, SOAP, JMS, POP3, IMAP, HTTP и TCP. В JMeter реализованы возможность создания большого количества запросов с помощью нескольких компьютеров при управлении этим процессом

с одного из них, логирование результатов тестов и разнообразная визуализация результатов в виде диаграмм, таблиц и т. п.

Ключевое понятие в JMeter —- план тестирования. План тестирования приложения представляет собой описание последовательности шагов, которые будет исполнять JMeter. План тестирования может содержать:

а) Группы потоков (Thread Groups) — элемент, позволяющий конфигурировать многопоточный запуск тестов

б) Контроллеры — позволяют создаватьтесты со сложной логической структурой.

в) Слушатели — отображают результаты выполнения тестов

г) Соответствия — позволяют сравнивать полученные результаты с ожиданиями пользователя

д) Сэмплеры — основные элемнты тест-плана, в которых формируется тело запроса, тестовый шаг.

На данный момент в JMeter нет отдельных средств для тестирования Flex приложений, однако есть возможность записи http запросов, содержащих в себе тело AMF сообщения, с помощью прокси-сервера. Далее перехваченные запросы могут быть перенесены в план тестирования и запущены в режиме нагрузочного тестирования. Созданные в JMeter тесты могут быть запущены с помощью систем сборок проектов Maven и Ant. Также JMeter свободно интегрируется со многими серверами сборок, такими как Jenkins и Hudson.

JMeter является бесплатным кросс-платформенным Java-приложением с открытым исходным кодом и удобным API. Существует большое количество плагинов к JMeter, значительно расщиряющих его базовую функциональность.

1.3 Итоги

В предыдущем разделе был рассмотрен ряд инструментов для тестировани Flex приложений. Для наглядности и простоты анализа все

исследованные продукты были собраны в таблицу и охарактеризованы по следующему набору критериев:

а) Возможность проведения нагрузочных тестов;
б) Поддержка протокола AMF.
в) Простота создания тестовых сценариев.
г) Возможность запуска тестов из командной строки.
д) Кроссплатформенность;
е) Условия распростронения продукта;
ж) Документация.

Для каждого критерия (в таблице указаны их номера), была дана оценка по пятибальной шкале, характеризующая степень того, в какой мере тот или иной продукт удовлетворяет данному критерию.

Таблица 1.1 — Пример короткой таблицы с длинным названием на много длинных-длинных строк

Инструмент тестирования	1	2	3	4	5	6	7
HP QuickTest Professional	2	0	3	4	0	1	4
IBM Rational Functional Tester	2	0	4	3	5	1	4
NeoLoad	5	5	4	4	5	1	3
Apache JMeter	5	3	4	5	5	5	4

Из таблицы видно, что по обозначенным критериям наиболее подходящим для решения поставленной задачи является инструмент тестирования Apache JMeter, решающим фактором стало условие распространения продукта с открытым исходным кодом, а также его популярность. Для того, чтобы JMeter мог обеспечивать полную поддержку тестирования Flex приложений, необходимо будет расширить его фунционал, добавив отображение AMF сообщений в понятной пользователю форме и возможность запуска тестов в несколько потоков без дублирования сессий с сервером.

2 Разработка модуля

2.1 Анализ требований к модулю

2.1.1 Требования

На основании приведённых ранее целей и задач проекта были разработаны функциональны требования. Для наглядности и удобства они были сведены в таблицу. Для классификации требований были выбраны следующие критерии:

а) Приоритет — критерий оценки полезности реализации требования для конечного пользователя, важности для достижения поставленных перед проектом целей.

б) Трудоёмкость — критерий отображает предварительную оценку сложности реализации требования, количество привлекаемых для этого ресурсов.

в) Риск — интегральный критерий, введением которого предпринимается попытка оценить во-первых возможность невыполнения требования, во-вторых возможную ошибку в оценке по двум предыдущим критериям (в первую очередь трудоёмкости).

Для каждого из критериев введена шкала из трёх уровней: низкий, средний, высокий. Размерность шкалы выбрана минимальной исходя из соображений простоты и наглядности. Так как размер проекта и количество предъявляемых к нему требований незначительны, то таких шкал вполне достаточно для исчерпывающей классификации требований а также принятия проектных и организационных решений.

2.2 Выбор технического средства решения задачи

2.3 Архитектура модуля

2.4 Реализация

2.5 Руководство пользователя

2.5.1 Установка JMeter

Для начала необходимо скачать с сайта производителя zip архив, содержащий все необходимы для установки файлы, а затем распаковать его на диске. Место установки JMeter далее будем называть JMETER_HOME.

2.5.2 Установка модуля amf-translator

Чтобы подключить к JMeter модуль amf-translator, достаточно добавить jar-файл приложения в каталог JMETER_HOME/lib/ext.

2.5.3 Запуск приложения

Приложение запускается с помощью файла jmeter.bat или ApacheJMeter.jar, находящихся в каталоге JMETER_HOME/bin.

2.5.4 Настройка прокси-сервера

После запуска в окне приложения с левой стороны нам доступно дерево элементов. Чтобы создать прокси сервер с поддержкой протокола AMF, необходимо правой кнопкой мыши кликнуть по элементу WorkBench, а затем добавить элемент AMF Proxy Server (WorkBench > Add > Non-Test Elements > AMF Proxy Server).

В поле AmfProxy Port необходимо указать номер порта, который будет слушать наш прокси сервер. Если указать, например, 9090, то прокси-сервер будет запущен на localhost:9090. Затем точно такие же настройки прокси-сервера устанавливаются в браузере, с помощью которого будет производиться тестирование. Также стоит убедиться, что указанный Вами порт уже не занят другим приложением.

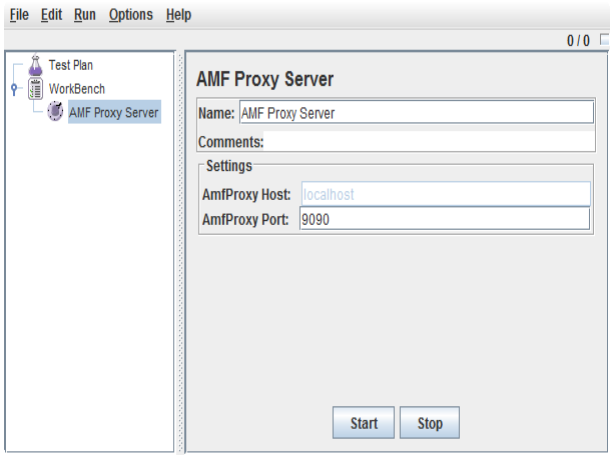

Рисунок 2.1 — Настройка прокси-сервера

2.5.5 Запись тестового сценария

После того, как в AMF Proxy Server установлены все необходимые параметры, нажимается кнопка Start, запускающая прокси-сервер.

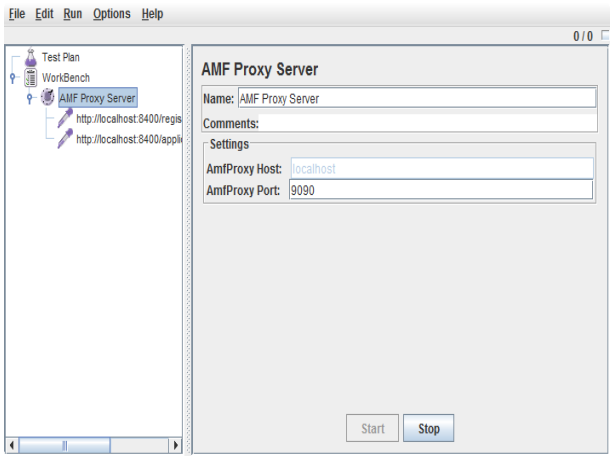

Рисунок 2.2 — Запуск прокси-сервера

Затем тестируемое приложение открывается в браузере, для которого также применены соответсвующие настройки прокси-сервера, и пользователь может выполнять с Flex приложением необходимые операции, которые будут записываться AMF Proxy Server в виде элементов AMF RPC Sampler и в дальнейшем могут быть перенесены в тест-план. Чтобы завершить запись тестовых запросов, необходимо нажать кнопку Stop. После завершения записи тестов, все перехваченные запросы отображаются в дереве элементов JMeter в качестве дочерних элементов AMF Proxy Server.

2.5.6 Создание тест-плана

Создание тест-плана в JMeter осуществляется следующим образом. В первую очередь добавляется группа потоков - Thread Group (Test Plan > Threads (Users) > Thread Group).

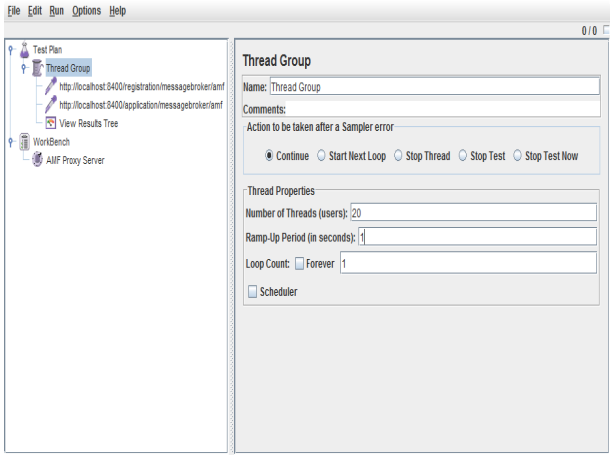

Рисунок 2.3 — Создание тест-плана

Этот элемент является ключевым в тест-плане JMeter, именно его функционал отвечает за реализацию нагрузочного тестирования — многопоточного запуска последовательности тестовых шагов. В данном элементе задается следующее:

а) Действия, которые будут производиться в случае, если в тест выполняется с ошибкой (Action to be taken after a Sampler error);

б) Число потоков, в которое будут запускаться шаги тест-плана (Number of Threads);

в) Интервал, в течение которого будет запущено указанное в предыдущем параметре число потоков (Ramp-Up Period);

г) Число повторений набора тестов (Loop Count);

д) Расписание запуска тестов (Scheduler).

Затем в Thread Group в качестве дочерних элементов добавляются шаги тестов, которые будут запускаться с указанными характеристиками. В нашем случае мы переносим элементы AMF RPC Sampler, записанные с помощью прокси. Помимо этого следует добавить визуалайзер результатов, чтобы иметь возможность отслеживать ход тестового сценария (Thread Group > Add > Listener). JMeter предлагает большой

выбор таких элементов, для примера будем использовать View Results Tree. После того как план сформирован, он может быть сохранён. (File > Save Test Plan As...)

2.5.7 Запуск тестов

Чтобы запустить содержимое элемента Test Plan, необходимо выбрать в основном меню Run > Start. После заврешения прогона тестов результаты их выполнения можно наблюдать в View Results Tree.

2.6 Методика тестирования