# Модуль JMeter для имитации работы GWT клиента.

Оглавление

[Модуль JMeter для имитации работы GWT клиента. 1](#_Toc423109523)

[Общие положения. 1](#_Toc423109524)

[Модуль Proxy сервер. 1](#_Toc423109525)

[Библиотека, для установки в jmeter 2](#_Toc423109526)

## Общие положения.

Модуль состоит из двух частей.

Первая часть – это Proxy сервер, который отлавливает запросы от клиента GWT на сервер и сохраняет их в специальном файле. При завершение работы Proxy сервера комбинацией клавиш Contrl+C формируется Jmeter скрипт, содержащий в себе все отловленные вызовы.

Вторая часть – это классы, работающие в среде JMeter, позволяющие выполнить записанный с помощью Proxy сервера скрипт. На стороне JMeter можно в широких рамках модифицировать запрос на сервер и анализировать ответы от сервера сохраняя в переменных объекты, которые могут. Это возможно благодаря наличию десериализатора и сериализатора запросов и ответов из формата GWT-RPC в java объекты, с которыми можно работать из Самплеров BeanShell на языке java.

Код модуля (обе часи) располагается в проекте CM5SRC\CM5\Performance\gwt-rpc-proxy. Проект собирает 2 jar файла. Первый gwt-rpc-proxy-X.X.X-X-SNAPSHOT-jar-with-dependencies.jar – для запуска proxy, второй gwt-rpc-proxy-X.X.X-X-SNAPSHOT-jar-with-dependencies.jar – для копирования в папку /lib/ext в jmeter.

## Модуль Proxy сервер.

Proxy сервер это консольное приложение написанное на языке java. Запускается командой

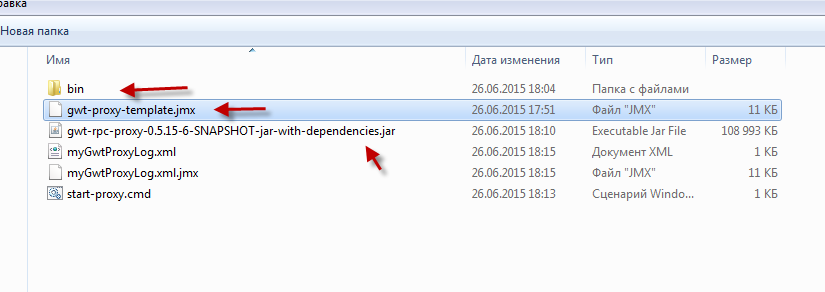
"%JAVA\_HOME%\bin\java" -cp gwt-rpc-proxy-0.5.15-6-SNAPSHOT-jar-with-dependencies.jar ru.intertrust.performance.gwtrpcproxy.GwtRpcProxy http://localhost:8080 8090 "myGwtProxyLog.xml"

<http://localhost:8080> – адрес AF5 сервера

8090 – Порт на котором работает Proxy

myGwtProxyLog.xml – Файл, в который сохраняется журнал взаимодействия клиента и сервера

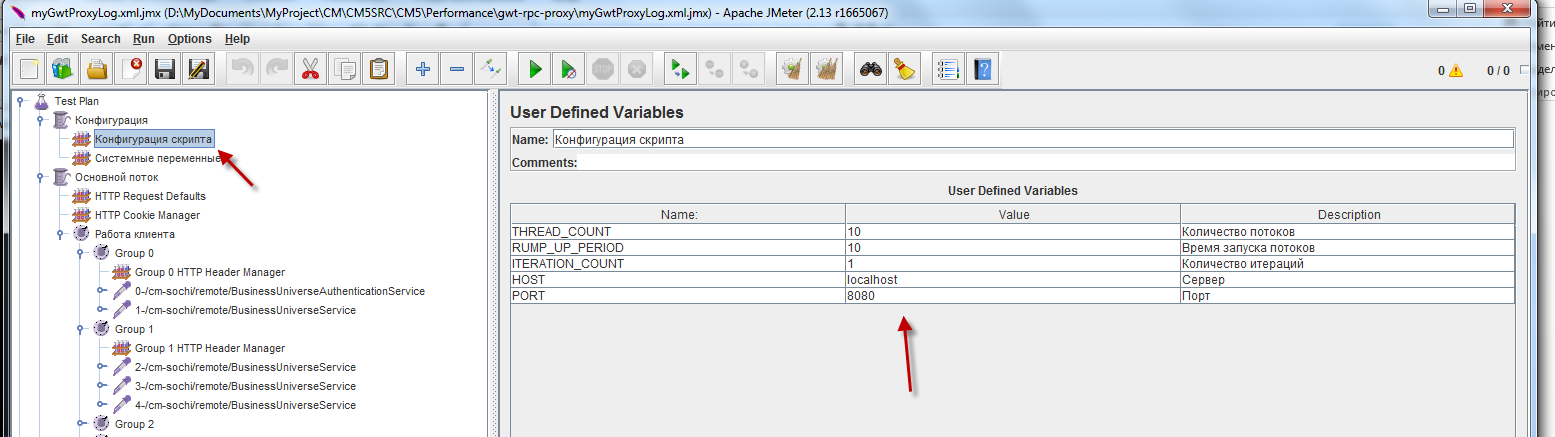
В папке в jar файлом обязательно должен лежать файл gwt-proxy-template.jmx и директория bin из каталога проекта.



После запуска прокси, для записи скрипта необходимо обратится браузером по адресу где запущен proxy и указать порт на котором запущен прокси (8090 из примера запуска), например сервер af5 запущен на <http://cm45.inttrust.ru:8080/ssrv-war> а прокси запускается на локальной машине. В этом случае переметры запуска должны быть http://cm45.inttrust.ru:8080 8090 "myGwtProxyLog.xml", а браузером обращаться по адресу <http://localhost:8090/ssrv-war>. Далее выполняем те действия, которые попадут в скрипт. Создаем документы, листаем списки итд. Запись должна обязательно начинаться с аутентификации в окне логина. После окончания сценария в окне proxy сервера нажмите комбинацию contr-C при этом proxy остановится а в файл указанный в параметрах сохранится журнал взаимодействия в виде xml файла а так же сформируется jmeter скрипт с тем же именем что и xml файл с припиской расширения .jmx, готовый для загрузки в среде jmeter. Все действия пользователя в среде jmeter будут разделены на группы. Разделение на группы производится автоматически по времени. Если пользователь простаивал более секунды, создастся новая группа. Группы облегчают дальнейшую работу с сгенеренным скриптом.

## Библиотека, для установки в jmeter

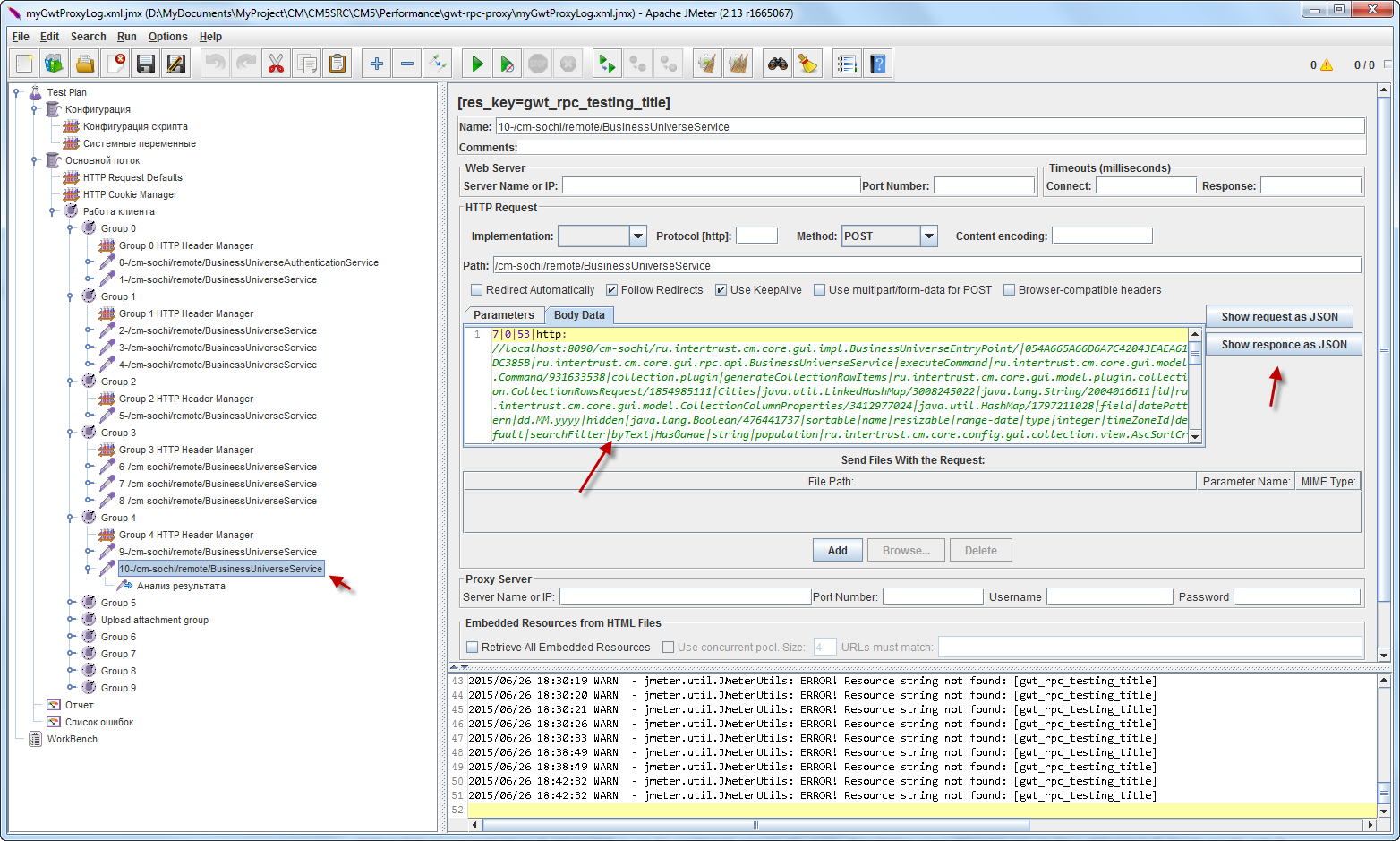
Для работы скрипта необходимо скопировать библиотеку gwt-rpc-proxy-X.X.X-X-SNAPSHOT.jar в директорию %JMETER\_ HOME%\lib\ext. После можно открыть сгенеренный с помощью proxy скрипт. В открывшемся дереве тестового скрипта есть 2 области, обрасть один – конфигурационная.



Здесь указывается хост и порт сервера AF5 на который будет обращатся jmeter, а так же количество потоков, время запуска потоков и количество итераций скрипта.

В разделе Основной поток сохранены обращения клиента на сервер. При запуске скрипта в неизменном виде на сервер пойдут копии обращений, которые выполнялись при записи скрипта с помощью proxy сервера. Некоторые запросы не требуют модификации (например аутентификация, обновление списка, переключение по меню), а некоторые требуют модификации запросов, из за того что состояние базы на которой записывались скрипты отличается от базы на которой запускается тест. Так же при создание документов создаются объекты с другими идентификаторами. Отличными от тех которые были при записи скрипта.

Наш механизм позволяет из ответа сервера получить какие то данные, сохранить их в переменных скрипта и далее подменить запрос к серверу, заменив например поля в форме. Рассмотрим единичный сформированный запрос к серверу.



В данном запросе на клиенте вызвался выпадающий список с городами. Proxy записал запрос и ответ в поля семплера, но запрос в формате GWT-RPC не читаемый, и для облегчения работы на форме добавлены 2 кнопки Show request as JSON и Show response as JSON. При нажатие на эти кнопки отобразится форма с текстовым полем, где можно посмотреть текст запроса или ответа в формате json. Эти представления сильно облегчают написание скриптов. Далее при генерации как дочерний элемент к элементу запроса к серверу автоматически добавлен семплер пост процессора, который вызывается после получения результата от сервера. В этом скрипте уже добавлен код проверки на возникновение ошибки. Разработчик скрипта может добавить свой код в этот скрипт. Например для сохранения произвольной записи из выпадающего списка надо добавить следующие строки:

ru.intertrust.cm.core.gui.model.form.widget.CollectionRowsResponse responce = GwtUtil.decodeResponce(prev);

//Запоминаем случайный город

ru.intertrust.cm.core.business.api.dto.Id city = GwtUtil.getRndCollectionsRow(responce);

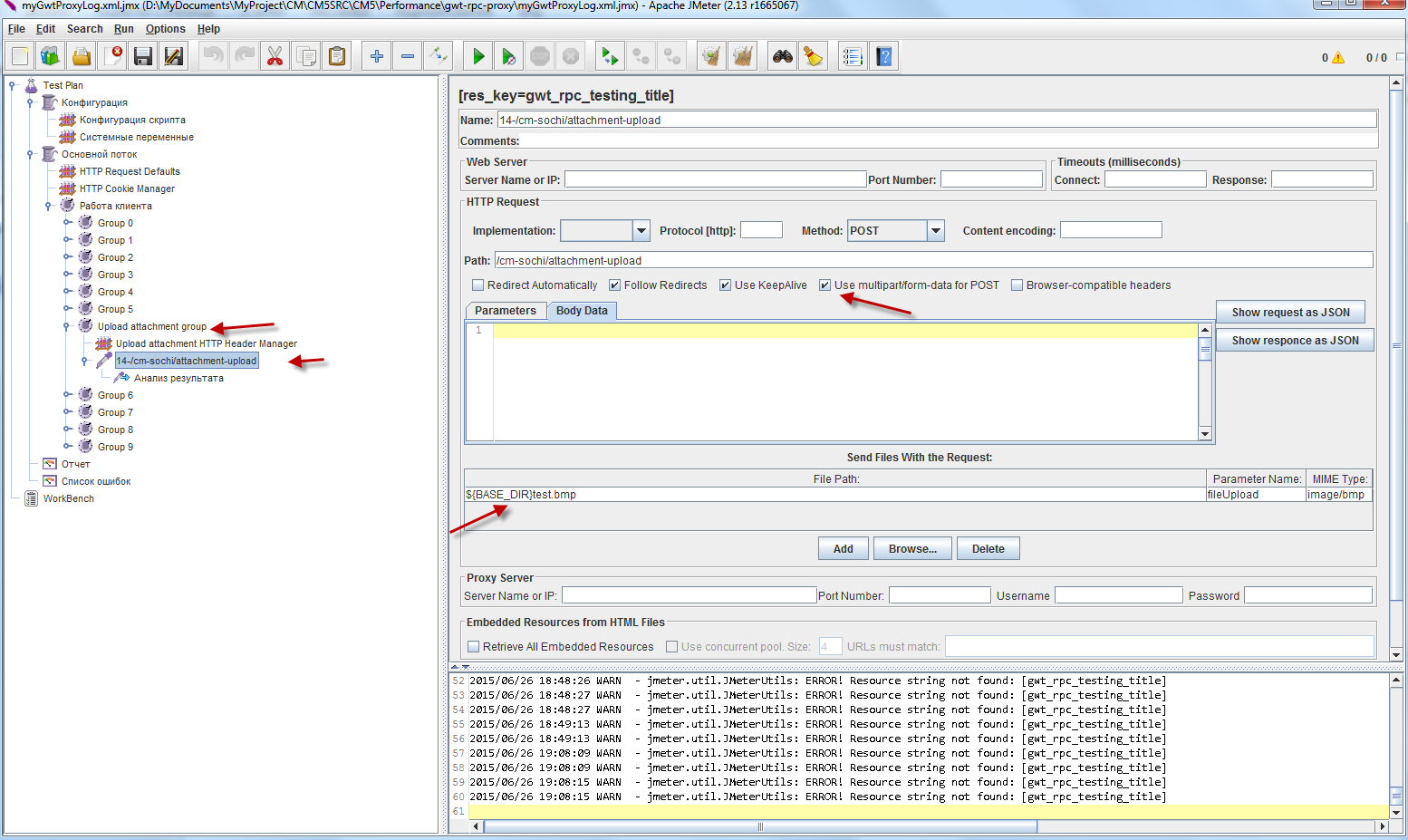
vars.putObject("RND\_CITY", city);

Метод GwtUtil.decodeResponce декодирует ответ в java объект

Метод GwtUtil.getRndCollectionsRow получает произвольную строку

Метод vars.putObject("RND\_CITY", city) сохраняет полученный идентификатор в переменной, дальнейшие скрипты могут ее использовать.

Обособленно стоит работа с вложениями. При отправки вложения на сервер, например при добавления документа к карточке, proxy сервер сформирует специальную группу, в которую добавит единственный запрос загрузки файла, а файл который отправляли при записи скрипта сохранит в текущей директории с proxy сервером. В самплер отправки файла пропишется автоматически путь к файлу.



Так же сформируется несколько измененный пост процессор, который уже содержит код получения имени временного файла на сервере и сохранение его в переменной ATTACHMENT\_TEMP\_NAME. Это имя надо подставить в значение виджета AttachmentBoxState при сохранение формы документа, к которой был присоединён файл, иначе на сервере будут происходить ошибки отсутствия файла.

Для модификации запроса, который идет на сервер (например подстановки произвольного имени, а так же подстановки случайного идентификатора) необходимо найти запрос, который формируется при нажатие на кнопку Сохранить. Ориентиром может служить строка save.action в тексте данных отправляемых на сервер.

Далее надо добавить BeanShell PreProcessor в качестве дочернего элемента к семплеру запроса на сервер. В этом пре процессоре можно изменить сохраненный с помощью proxy сервера запрос, для того чтобы, например, не возникало ошибок неуникальности имени документа (если этого не сделать то сохраненный запрос будет посылать на сервер запросы создания совершенно одинаковых документов, копии того, что был создан при записи скрипта).

Добавим в код препроцессора код:

import ru.intertrust.performance.jmetertools.\*;

//Декодируем запрос

GwtRpcRequest request = GwtUtil.decodeRequest(sampler);

//Получаем виджет с именем

ru.intertrust.cm.core.gui.model.form.widget.TextState nameState = GwtUtil.getWidgetState(request, "2");

//Устанавливаем новое значение

nameState.setText(GwtUtil.getRndString(20));

// получаем виджет country\_attachment

ru.intertrust.cm.core.gui.model.form.widget.AttachmentBoxState countryAttachmentState = GwtUtil.getWidgetState(request, "country\_attachment");

//Устанавливаем новое значение

countryAttachmentState.getAttachments().get(0).setTemporaryName(vars.get("ATTACHMENT\_TEMP\_NAME"));

// получаем виджет capital (на форме 8a)

ru.intertrust.cm.core.gui.model.form.widget.ListBoxState capital = GwtUtil.getWidgetState(request, "8a");

//Устанавливаем новое значение, ранее сохраненный город в 10-/cm-sochi/remote/BusinessUniverseService

capital.getSelectedIds().get(0).clear();

capital.getSelectedIds().get(0).add(vars.getObject("RND\_CITY"));

//Кодируем измененный запрос и записываем в семплер

GwtUtil.setRequest(sampler, request);

Метод GwtUtil.decodeRequest декодирует запрос в объект GwtRpcRequest, из него мы потом сможем получить параметры запроса

Метод GwtUtil.getWidgetState получает статусы виджетов формы по имени виджета в xml конфигурации формы, далее значения статусов можно поменять, напимер в текстовом виджете установить новый текст nameState.setText

Метод GwtUtil.getRndString(20) генерит произвольную строку, длиной 20 символов

Метод vars.getObject("RND\_CITY") получает ранее сохраненный случайный идентификатор из выпадающего списка городов

Метод GwtUtil.setRequest кодирует измененный запрос в GWT-RPC вызов и копирует его в сэмплер, который отправит измененный запрос на сервер.

В строке countryAttachmentState.getAttachments().get(0).setTemporaryName(vars.get("ATTACHMENT\_TEMP\_NAME")); мы устанавливаем имя файла, загруженного на сервер ранее.

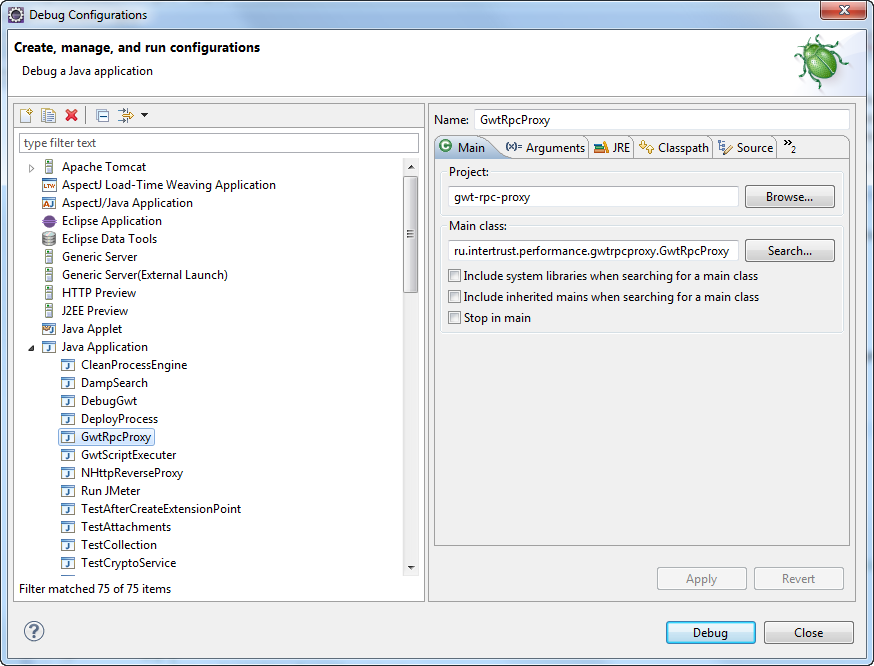
Возможности модификации не ограниченны приведенными примерами. Аналогичным образом можно выполнять практически любые модификации отправляемых запросов на сервер.

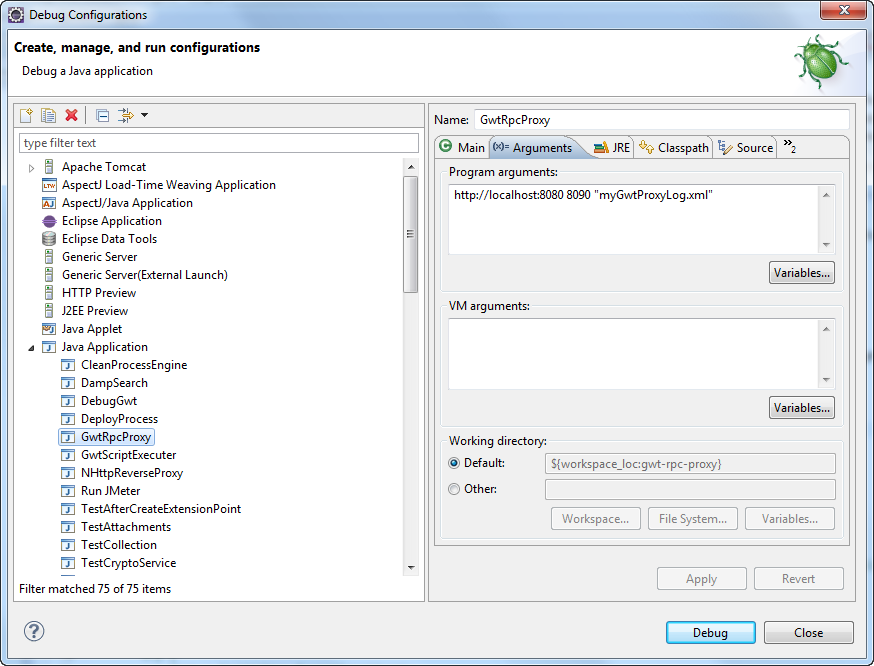
## Запуск

Пока не разработаны нормальные скрипты запуска proxy и jmeter, поэтому запускать надо из среды разработки, при этом подтянутся все нужные зависимости.

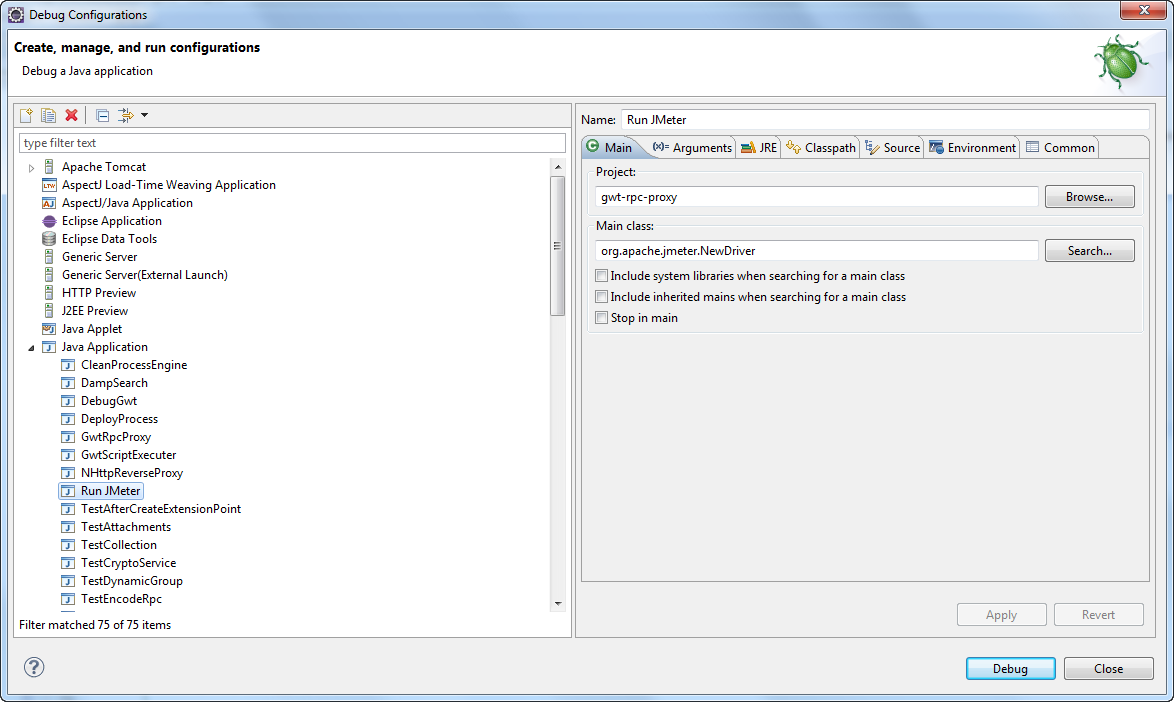
Скрины для эклипсы:

Запуск прокси





Запуск JMeter



В ближайшее время скрипты запуска прокси и jmeter без эклипсы будут разработаны.