Масштабируемая отказоустойчивая архитектура на основе серверов Apache HTTPD и Tomcat

Оглавление

[Масштабируемая отказоустойчивая архитектура на основе серверов Apache HTTPD и Tomcat 1](#_Toc465668832)

[1. Термины и сокращения 4](#_Toc465668833)

[2. Введение 4](#_Toc465668834)

[3. Установка HTTPD 5](#_Toc465668835)

[3.1 Проверка работы 6](#_Toc465668836)

[3.2 Создание Windows-сервиса для HTTPD 6](#_Toc465668837)

[3.3 Удаление Windows-сервиса HTTPD 7](#_Toc465668838)

[4. Установка Tomcat 7](#_Toc465668839)

[4.1 Настройки Tomcat 7](#_Toc465668840)

[4.1.1 Базовые настройки 7](#_Toc465668841)

[4.1.2 Особенности настройки Tomcat в Windows 8](#_Toc465668842)

[4.2 Создание Windows-сервиса для Tomcat 9](#_Toc465668843)

[4.3 Удаление Windows-сервиса Tomcat 9](#_Toc465668844)

[5. Настройка взаимодействия HTTPD-Tomcat 10](#_Toc465668845)

[5.1 Настройка на стороне HTTPD 10](#_Toc465668846)

[5.1.1 Настройка workers.properties 10](#_Toc465668847)

[5.1.2 Настройка mod\_jk.conf 10](#_Toc465668848)

[5.1.3 Настройка httpd.conf 10](#_Toc465668849)

[5.1.4 Коннектор mod\_jk 11](#_Toc465668850)

[5.2 Настройка на стороне Tomcat 11](#_Toc465668851)

[5.2.1 Настройка server.xml 11](#_Toc465668852)

[5.2.2 Коннектор tomcat-jk2 11](#_Toc465668853)

[5.3 Проверка работоспособности 11](#_Toc465668854)

[6. Настройки для приложений 12](#_Toc465668855)

[6.1 Настройка источников данных и пулов соединений 12](#_Toc465668856)

[7. Конфигурирование SSO в экземплярах Tomcat 13](#_Toc465668857)

[7.1 Встраивание собственного Login-модуля 13](#_Toc465668858)

[7.1.1 Файл context.xml приложения 13](#_Toc465668859)

[7.1.2 Файл определения JAAS Login-модуля 13](#_Toc465668860)

[7.2 Включение режима SSO в экземпляре Tomcat 14](#_Toc465668861)

[7.3 Настройки Windows-сервиса Tomcat 14](#_Toc465668862)

[8. Установка приложений JepRia на Tomcat 15](#_Toc465668863)

[8.1 Создание администратора Tomcat 15](#_Toc465668864)

[8.2 Автоматизированная установка приложений с помощью Ant 15](#_Toc465668865)

[8.3 Используемые библиотеки 15](#_Toc465668866)

[8.4 Используемые переменные 16](#_Toc465668867)

[9. Обеспечение отказоустойчивости серверов Tomcat 16](#_Toc465668868)

[9.1 Настройка восстановления работы Windows-сервиса Tomcat 16](#_Toc465668869)

[10. Нагрузочное тестирование приложений JepRia 19](#_Toc465668870)

[*10.1* Инструмент нагрузочного тестирования *Apache JMeter* 19](#_Toc465668871)

[10.2 Поддержка пользовательских сессий 19](#_Toc465668872)

[10.3 Установка JMeter 20](#_Toc465668873)

[10.4 Запуск JMeter 20](#_Toc465668874)

[10.5 Создание тест-плана 20](#_Toc465668875)

[10.5.1 Подключение модуля поддержки пользовательских сессий - “HTTP Cookie Manager” 22](#_Toc465668876)

[10.5.2 Подключение модулей просмотра и анализа результатов работы 24](#_Toc465668877)

[10.6 Создание узлов тестовых сценариев 28](#_Toc465668878)

[10.6.1 Создание узла сценария-инициализации “Login” 28](#_Toc465668879)

[10.6.2 Создание узла основого сценария 29](#_Toc465668880)

[10.6.3 Создание узла заключительного сценария “Logout” 31](#_Toc465668881)

[10.7 Запись тестовых сценариев 32](#_Toc465668882)

[10.7.1 Подключение модуля “HTTP(S) Test Script Recorder” 32](#_Toc465668883)

[10.7.2 Запись сценария инициализации “Login” 34](#_Toc465668884)

[10.7.3 Запись основного сценария 36](#_Toc465668885)

[10.7.4 Запись сценария “Logout” 36](#_Toc465668886)

[10.7.5 Проверка тестовых сценариев 36](#_Toc465668887)

[10.8 Настройка основного сценария 37](#_Toc465668888)

[10.8.1 Модуль контроля интенсивности “Constant Throughput Timer” 38](#_Toc465668889)

[10.9 Запуск теста 40](#_Toc465668890)

[10.10 Просмотр результатов работы теста 41](#_Toc465668891)

[11. Дистрибутивы, библиотеки 42](#_Toc465668892)

[11.1 Дистрибутив Apache Tomcat 43](#_Toc465668893)

[11.2 Коннектор Tomcat-JK2 43](#_Toc465668894)

[11.3 Дистрибутив Apache HTTPD 43](#_Toc465668895)

[11.4 Коннектор MOD\_JK 43](#_Toc465668896)

[11.5 Дистрибутив Apache JMeter 43](#_Toc465668897)

[12. Приложение 1. Настройка HTTPD. 43](#_Toc465668898)

[12.1 Настройка кэширования 43](#_Toc465668899)

[12.2 Настройки, связанные с безопасностью сервера приложений 44](#_Toc465668900)

[12.2.1 Скрытие названия и версии сервера приложений 44](#_Toc465668901)

[13. Приложение 2.Пример настройки файла *server.xml* в Tomcat. 44](#_Toc465668902)

[14. Приложение 3.Пример настройки файла mod\_jk.conf 45](#_Toc465668903)

[15. Приложение 4 46](#_Toc465668904)

[16. Приложение 5. Конфигурирование Tomcat для отладки в Eclipse 46](#_Toc465668905)

[17. Ссылки 53](#_Toc465668906)

# Термины и сокращения

*HTTPD (Hyper Text Transfer Protocol Daemon)* – Apache HTTP Server;

*Вертикальное масштабирование* – увеличение производительности системы при увеличении ресурсов вычислительного узла;

*Горизонтальное масштабирование* – увеличение производительности системы при увеличении количества вычислительных узлов;

*Reverse Proxy (Обратный прокси)* - тип прокси-сервера, который ретранслирует запросы клиентов из внешней сети на один или несколько серверов, логически расположенных во внутренней сети. При этом для клиента это выглядит так, будто запрашиваемые ресурсы находятся непосредственно на прокси-сервере.

*Realm* представляет собой «базу данных» пользовательских имён, паролей и ролей, используемых системами безопасности.

# Введение

Документ содержит основные положения по настройке системы серверов Apache HTTP Server (HTTPD) + Tomcat.



Рисунок 1.Общая схема взаимодействия серверов Apache HTTPD, Tomcat и СУБД

Архитектура предполагает простую и удобную возможность вертикального и горизонтального масштабирования функционала приложений, разрабатываемых в стандарте J2EE, при увеличении интенсивности запросов к системе и (или) увеличении одновременно работающего функционала.

Apache HTTPD выполняет функции:

* Reverse proxy для экземляров серверов Tomcat, обеспечивая обработку внешних запросов по стандартному порту 80;
* Балансировку нагрузки на экземпляры серверов Tomcat;
* Эффективный доступ к статическим ресурсам.

# Установка HTTPD

Используемый дистрибутив HTTPD [см. Дистрибутив Apache HTTPD] предназначен для работы в Windows.

Для установки его достаточно распаковать в одну из директорий, например, *с:\Apache24*.

Запуск сервера из командной строки выполняется командой:

*httpd.exe*

Запуск произойдет при распаковке сервера в корень диска. Если распаковать в поддиректории, то в конфигурационном файле (*%APACHE\_HOME%/conf/httpd.conf)* необходимо указать пути (параметры DocumentRoot, ServerRoot и т.д.).

При необходимости запуска сервера на ином порту (отличном от 80), в *%APACHE\_HOME%/conf/httpd.conf* , 80 порт изменяется на новый порт в следующих строчках:

Строчка 46: Listen 80

Строчка 172: ServerName localhost:80

## Проверка работы

При вводе в адресную строку браузера <http://localhost> он должен выдать такую страницу:



## Создание Windows-сервиса для HTTPD

Windows-сервис используемой версии HTTPD создаётся командой

***httpd -k install***,

которая запускается из директории *%APACHE\_HOME%/bin/*

Для созданного сервиса рекомендует установить автоматический запуск.

Для установки сервиса с конкретным именем необходимо использовать ключ –n.

Пример команды: httpd.exe -k install -n "MyServiceName"

## Удаление Windows-сервиса HTTPD

Для удаления сервиса Windows-сервиса HTTPD используется команда

***httpd -k uninstall***

Для удаления сервиса с конкретным именем необходимо использовать ключ –n.

Пример команды: httpd.exe -k uninstall -n "MyServiceName"

# Установка Tomcat

Используемый дистрибутив Tomcat [см.Дистрибутив Apache Tomcat] ориентирован на работу в 64-разрядной Windows, для её работы необходима соответствующая 64-разрядная версия Java 1.7+.

## Настройки Tomcat

Настройки Tomcat целесообразно разделить на базовые настройки, настройки для приложений и настройки взаимодействия с HTTPD (вынесены в раздел *Настройка взаимодействия HTTPD-Tomcat*).

### Базовые настройки

Базовые настройки включают настройки внешних переменных, настройки источников данных и пулов соединений.

#### Внешние переменные

Внешние переменные для Tomcat определяются в файле (его нужно создать) *%TOMCAT\_HOME%/bin/setEnv.bat* (здесь и далее используется нотация Windows), который автоматически вызывается в самом начале работы*.*

Для Tomcat используемой версии необходима версия Java 1.7+. Для неё в ***setEnv.bat*** задаётся переменная *JAVA\_HOME*, а директория *%JAVA\_HOME%/bin* прописывается в переменной *PATH*, например:

*set JAVA\_HOME=D:/Program Files/Java/jdk1.8.0\_45*

*set PATH=%JAVA\_HOME%/bin;% PATH%*

#### Проверка работы

После выполнения базовых настроек Tomcat можно проверить его работоспособность.

Запуск Tomcat из командной строки выполняется из директории *%TOMCAT\_HOME%/bin* командой:

*startup*

После запуска в DOS-окно запуска должно выглядеть примерно так:



При вводе в адресную строку браузера <http://localhost:8080> (8080 – порт по умолчанию) он должен выдать такую страницу:



### Особенности настройки Tomcat в Windows

При работе с Tomcat в Windows иногда возникает проблема во время undeploy, характеризующаяся ошибками вида:

FAIL - Unable to delete [F:\apache-tomcat-7.0.33\webapps\balaji]. The continued presence of this file may cause problems.

Данная проблема связана с блокировкой ресурсов во время deploy и решается модификацией файла conf/context.xml. К родительскому узлу <Context> необходимо добавить атрибут antiResourceLocking=”true”. В результате он будет выглядеть следующим образом:

<Context antiResourceLocking="true">

См. также: <http://stackoverflow.com/q/14873219>

## Создание Windows-сервиса для Tomcat

Для того, чтобы создать Windows-сервис для Tomcat, например, с именем *Tomcat\_1*, нужно перейти в директорию *<TomcatHome>/bin*,открыть в нём консольное окно и последовательно выполнить команды:

*set java\_home=<TomcatJavaHome>*

*service install Tomcat\_1*



Для созданного сервиса рекомендует установить автоматический запуск.

## Удаление Windows-сервиса Tomcat

Для удаления сервиса Tomcat используется команда

*service uninstall <serviceName>*

# Настройка взаимодействия HTTPD-Tomcat

Настройка взаимодействия HTTPD-Tomcat включает настройку на стороне HTTPD и настройку на стороне Tomcat.

## Настройка на стороне HTTPD

Настройка сервера HTTPD для взаимодействия с Tomcat включает:

* Настройку файла *%APACHE\_HOME%/conf/jk /workers.properties*
* Настройку файла *%APACHE\_HOME%/conf/jk /mod\_jk.conf*
* Настройку файла *%APACHE\_HOME%/conf/httpd.conf*
* Подключение коннектора *mod\_jk*

### Настройка workers.properties

Файл *workers.properties* описывает параметры экземпляров Tomcat, взаимодействующих с HTTPD. Параметры *port* и *host* каждого worker-а должны совпадать с соответствующими настройками Tomcat в *server.xml*. Ниже приводится пример файла *workers.properties*, в котором определяется два worker-а – ***worker1***и ***worker2***:

*# BEGIN workers.properties*

*worker.list=worker1,worker2*

*worker.worker1.port=8009*

*worker.worker1.host=localhost*

*worker.worker1.type=ajp13*

*worker.worker2.port=8109*

*worker.worker2.host=localhost*

*worker.worker2.type=ajp13*

*# END workers.properties*

### Настройка mod\_jk.conf

Файл *mod\_jk.conf* указывает группам ресурсов (приложениям) соответствующие им worker-ы.

Файл *mod\_jk.conf* создаётся в ручном режиме и размещается в директории *%APACHE\_HOME%/conf/jk*  (предлагаемый режим, когда *mod\_jk.conf* автоматически создаётся при запуске Tomcat, работает некорректно).

В п.*Приложение 4.Пример настройки файла mod\_jk.conf* приведён пример **ручной** настройки, обеспечивающей работу с встроенными примерами приложений Tomcat).

### Настройка httpd.conf

В конце файла *httpd.conf* необходимо подключить файл *mod\_jk.conf*:

…

*# Include mod\_jk's specific configuration file*

*Include conf/jk/mod\_jk.conf*

…

### Коннектор mod\_jk

*mod\_jk* – коннектор, поддерживающий взаимодействие Apache HTTPD-Tomcat по протоколу AJP со стороны HTTPD.

Она добавляется в директорию %APACHE\_HOME%/modules.

Скачать можно здесь [см.Коннектор MOD\_JK].

## Настройка на стороне Tomcat

Настройки взаимодействия Tomcat с HTTPD включают:

* настройку *server.xml*
* настройку коннектора Apache HTTPD-Tomcat

### Настройка server.xml

См.пример настройки в Приложение 3.Пример настройки файла *server.xml* в Tomcat.

### Коннектор tomcat-jk2

Tomcat-jk2 – библиотека (jar), реализующая коннектор, поддерживающий взаимодействие Apache HTTPD-Tomcat по протоколу AJP со стороны Tomcat.

Она добавляется в директорию %TOMCAT\_HOME%/lib.

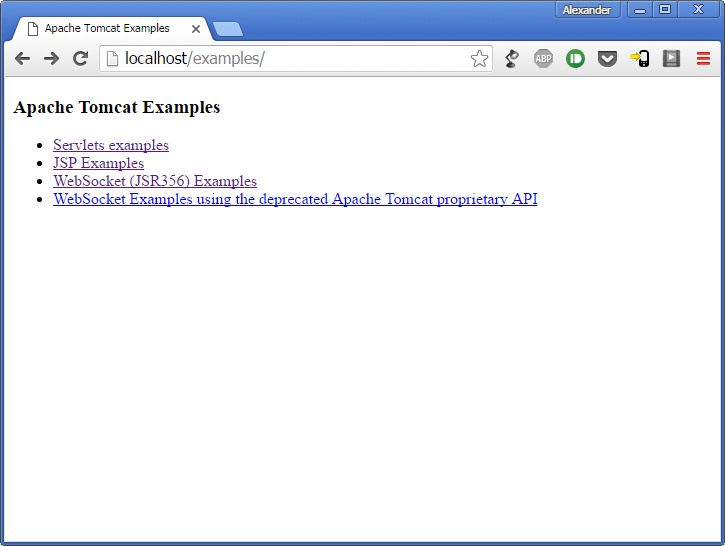
Скачать можно здесь [см. Коннектор Tomcat-JK2].

## Проверка работоспособности

После того, как настройки на обеих сторонах выполнены, для проверки работоспособности нужно:

1. Перезапустить сервис Tomcat
2. Перезапустить сервис HTTPD
3. В адресной строке браузера набрать <http://localhost/examples>,

После этого в окне браузера должна появиться страница:



# Настройки для приложений

Настройки для приложений включают настройки для источников данных и пулов соединений.

## Настройка источников данных и пулов соединений

Настройка пулов соединений для источников данных Tomcat выполняется в файле *%TOMCAT\_HOME%/conf/context.xml*.

Ниже приводится пример настройки пула соединений для источника данных: *jdbc/JEPDS* , настроенного на Oracle:

…

*<Context>*

*…*

*<Resource*

*name="jdbc/ JEPDS "*

*auth="Container"*

*type="oracle.jdbc.pool.OracleDataSource"*

*factory="oracle.jdbc.pool.OracleDataSourceFactory"*

*url="jdbc:oracle:thin:@//host:1521/DB1"*

*user="dbschema1"*

*password=" welcome "*

*connectionCachingEnabled="true"*

*/>*

*TODO разобраться с настройками производительности пула соединений Oracle*

*…*

*<Context>*, где

* *auth* – способ управления безопасностью – в данном случае – Container Managed Security.
* *maxActive* (int) – Максимальное число активных соединений одновременно взятых из пула. По умолчанию: 100. Рекомендуемое значение: 30.
* *maxIdle* (int) – Максимальное число соединений, находящихся в пуле. Рекомендуемое значение 30.
* *maxWait* (int) – Максимальное число миллисекунд, в течение которых пул будет ожидать освобождения соединения перед тем, как выкинуть exception. По умолчанию: 30000 (30 секунд).  
  TODO: Определить необходимое значение при загрузке больших файлов.
* *jmxEnabled* (boolean) Указывает необходимость поддержки пулом JMX. По умолчанию: true.

# Конфигурирование SSO в экземплярах Tomcat

Согласно инструкции Tomcat в режиме работы SSO один экземпляр Tomcat может поддерживать только один Realm (TODO в то же время, возможно, стоит проверить, может ли один Realm размещаться на нескольких экземплярах Tomcat).

Настоящий документ ориентирован на конфигурацию работы с несколькими Realm-ами, каждый из которых обслуживается отдельным экземпляром Tomcat.

## Встраивание собственного Login-модуля

Для встраивания в Tomcat кастомизированного Login-модуля необходимо создать два конфигурационных файла: файл определения Realm и файл JAAS Login.

### Файл context.xml приложения

Отнесение приложения к тому или иному realm-у определяется значением атрибута *appName* тега *Context* файла *META-INF/сontext.xml* приложения (не путать с файлом *%TOMCAT\_HOME%/conf/context.xml*) и имеет следующий вид:

*<?xml version="1.0" encoding="UTF-8"?>*

*<Context>*

*<Realm className="org.apache.catalina.realm.JAASRealm"*

***appName="<REALM\_ID>"***

*userClassNames="com.technology.jep.jepcommon.security.JepPrincipal"*

*roleClassNames="com.technology.jep.jepcommon.security.JepPrincipal" />*

*</Context>*

, где пользовательский и ролевой приципалы определяются классом

*com.technology.jep.jepcommon.security.JepPrincipal*.

Значение *<REALM\_ID>* в этом случае должно совпадать с идентификатором realm-а, определяемого в файле *jaas.conf* (см. 7.1.2).

### Файл определения JAAS Login-модуля

##### Конфигурационный файл JAAS Login *jaas.conf* размещается в директории *%CATALINA\_BASE%/conf* и имеет вид:

*/\*\**

*\* Login Configuration for JAAS*

*\*/*

*<REALM\_ID> {*

*com.technology.jep.jepcommon.security.tomcat.JepLoginModule sufficient dataSource="****< REALM\_DATASOURSE\_ID>";***

*};*

, где встраиваиваемый Login-модуль определяется классом

*com.technology.jep.jepcommon.security.tomcat.JepLoginModule*.

Каждому realm-у соответствует источник данных *REALM\_DATASOURSE\_ID*, который используется для аутентификации и авторизации пользователей.

Для приложений JepRia указанные выше идентификаторы *<REALM\_ID>* и *< REALM\_DATASOURSE\_ID>* принимают значение *RFInfoDS* и *jdbc/RFInfoDS* соответственно.

## Включение режима SSO в экземпляре Tomcat

Режим SSO в экземпляре Tomcat включается расскомментариванием в файле *server.xml* (см.п.12)

строки

*<Valve className="org.apache.catalina.authenticator.SingleSignOn" />*

## Настройки Windows-сервиса Tomcat

Настройки Windows-сервиса Tomcat выполняются в скрипте установки сервиса *service.bat*.

Каждый экземпляр Tomcat устанавливается как сервис Windows, как выше было сказано командой

*service install <TomcatServiceName>*

При этом перед установкой каждого сервиса в скрипт *service.bat* вносится два изменения:

* Переменной *%JAVA\_HOME%* присваивается значение пути того JDK, на котором будет выполняться сервис,
* В опцию *–JvmOptions* добавляется пара параметров:

*-Dorg.apache.catalina.authenticator.Constants.SSO\_SESSION\_COOKIE\_NAME= <REALM\_ID>\_JSESSIONIDSSO* и

*-Djava.security.auth.login.config=%CATALINA\_BASE%/conf/jaas.conf*

*<REALM\_ID>\_JSESSIONIDSSO –* идентификатор SSO-сессии для Realm-а устанавливаемого сервиса Tomcat (устанавливается в cookie браузера при входе в первое из приложений Realm-а).Здесь *REALM\_ID* – идентификатор Realm-а, поддерживаемого устанавливаемым сервисом Tomcat.

Например:

*…*

*"%EXECUTABLE%" //IS//%SERVICE\_NAME% ^*

*--Description "Apache Tomcat 7.0.68 Server - http://tomcat.apache.org/" ^*

*--DisplayName "%DISPLAYNAME%" ^*

*--Install "%EXECUTABLE%" ^*

*--LogPath "%CATALINA\_BASE%\logs" ^*

*--StdOutput auto ^*

*--StdError auto ^*

*--Classpath "%CLASSPATH%" ^*

*--Jvm "%JVM%" ^*

*--StartMode jvm ^*

*--StopMode jvm ^*

*--StartPath "%CATALINA\_HOME%" ^*

*--StopPath "%CATALINA\_HOME%" ^*

*--StartClass org.apache.catalina.startup.Bootstrap ^*

*--StopClass org.apache.catalina.startup.Bootstrap ^*

*--StartParams start ^*

*--StopParams stop ^*

*--JvmOptions "****-Dorg.apache.catalina.authenticator.Constants.SSO\_SESSION\_COOKIE\_NAME=<REALM\_ID>\_JSESSIONIDSSO;-Djava.security.auth.login.config=%CATALINA\_BASE%/conf/jaas.conf****;-Dcatalina.home=%CATALINA\_HOME%;-Dcatalina.base=%CATALINA\_BASE%;-Djava.endorsed.dirs=%CATALINA\_HOME%\endorsed;-Djava.io.tmpdir=%CATALINA\_BASE%\temp;-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager;-Djava.util.logging.config.file=%CATALINA\_BASE%\conf\logging.properties" ^*

*--JvmMs 128 ^*

*--JvmMx 256*

*…*

На всякий случай отметим, что любые изменения в скрипте установки сервиса *service.bat* вступают в силу только после последующей переустановки сервиса парой команд:

*service uninstall <serviceName>*

*service install <serviceName>*

# Установка приложений JepRia на Tomcat

Web-приложения, оформленные war-файлом или структурированные согласно стандарту J2EE, устанавливаются на Tomcat простым копированием в директорию *%TOMCAT\_HOME%/webapps*.

## Создание администратора Tomcat

Автоматизированная установка выполняется от имени специального пользователя - администратора Tomcat, которого необходимо создать, определив его в файле *%TOMCAT\_HOME%/config/tomcat-users.xml* следующим образом (пароль, разумеется, вариабелен):

*<?xml version='1.0' encoding='utf-8'?>*

*<tomcat-users>*

*<role rolename="manager-gui"/>*

*<role rolename="manager-script"/>*

*<role rolename="manager-jmx"/>*

*<role rolename="manager-status"/>*

*<user username="admin" password="admin" roles="manager-gui,manager-script,manager-jmx,manager-status"/>*

*</tomcat-users>*

## Автоматизированная установка приложений с помощью Ant

Скрипт сборки приложений JepRia – *build.xml* – позволяет выполнять автоматизированную установку/удаление (deploy/undeploy) приложений на Tomcat средствами Ant при помощи целей (targets) *tomcat.deploy* и *tomcat.undeploy*.

## Используемые библиотеки

При установке/удалении приложений используются библиотеки:

*%TOMCAT\_HOME%/lib/catalina-ant.jar*

*%TOMCAT\_HOME%/lib/tomcat-coyote.jar*

*%TOMCAT\_HOME%/lib/tomcat-util.jar*

Сборщик Ant при установке приложения JepRia использует данные библиотеки из бинарного репозитория.

В процессе настройки автоматизированной установки необходимо убедиться, чтобы директория *%ANT\_HOME%/lib* не содержала иных версий этих библиотек.

## Используемые переменные

В файле *deploy.properties* необходимо определить переменные *DEPLOYMENT\_PATH, LOGIN, PASSWORD*, например:

…

*# Параметры установки для локального Tomcat*

*DEPLOYMENT\_PATH=http://localhost:8080/manager/text*

*LOGIN=admin*

*PASSWORD=admin*

…

# Обеспечение отказоустойчивости серверов Tomcat

Под отказоустойчивостью серверов Tomcat будем понимать возможность их перезапуска в случаях, когда по тем или иным причинам они прекращают свою работу.

Отказоустойчивость серверов Tomcat может обеспечиваться разными средствами. Например, они могут быть платформенно-независимыми и платформенно-ориентированными. К платформенно-независимым относятся такие продукты, как [Apache Commons Daemon](https://commons.apache.org/proper/commons-daemon/index.html) (для Windows есть специальное решение - [procrun](https://commons.apache.org/proper/commons-daemon/procrun.html)), [JSW](http://www.tanukisoftware.com/en/index.php), [YAJSW](http://yajsw.sourceforge.net/).

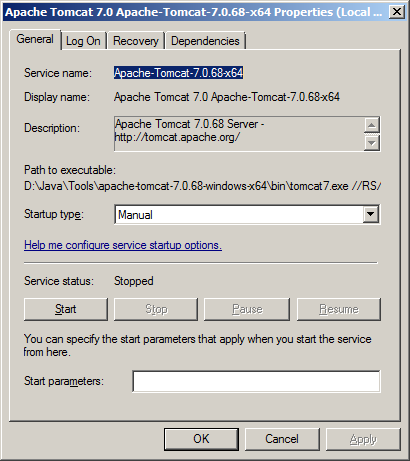
Из платформенно-ориентированных средств наиболее известны: для Linux - [MONIT](https://mmonit.com/monit/), для Windows – средства автоматического восстановления работы сервисов (в тех случаях, когда Tomcat установлен, как сервис Windows).

## Настройка восстановления работы Windows-сервиса Tomcat

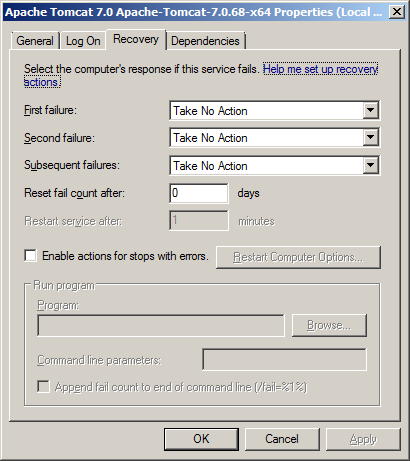
В рамках настоящей архитектуры для обеспечения отказоустойчивости Tomcat, установленных как Windows-сервисы, используется решение, предлагаемое в Professional Apache Tomcat 6 - автоматическое восстановление работы сервисов Windows.

Настройка выполняется следующей последовательностью действий:

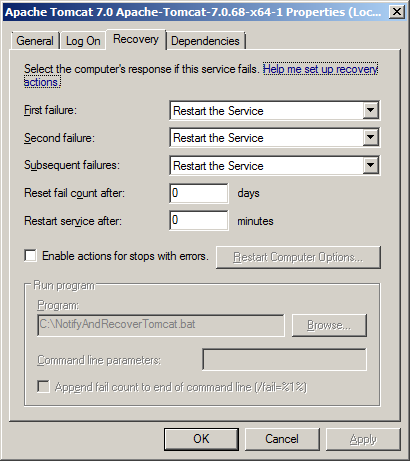
Открыть окно свойств сервиса Tomcat:



Перейти на вкладку *Recovery*:



Для обеспечения немедленного рестарта сервиса сразу после падения установить следующие настройки:



# Нагрузочное тестирование приложений JepRia

Нагрузочное тестирование установки используется для определения её возможности выдерживать различные нагрузки, для выявления её узких мест с целью выполнения настроек, которые бы обеспечивали необходимые параметры функционирования установки в промышленном окружении в боевом режиме.

## Инструмент нагрузочного тестирования *Apache JMeter*

*Apache JMeter* (см. п.6) является открытым и бесплатным программным продуктом, разработанным на Java, который в первую очередь предназначен для тестирования производительности Web-приложений.

При помощи Apache JMeter можно удобно создавать сценарии работы с приложениями, моделирующие действия пользователя, в том числе, посредством записи действий пользователя, чтобы проверять поведение приложения при запуске этих сценариев в различных режимах.

## Поддержка пользовательских сессий

Поддержка пользовательских сессий необходима в тех случаях, когда приложение разграничивает доступ к своим ресурсам в зависимости от прав, имеющихся у пользователя. В этих случаях в начале работы пользователь обычно проходит процедуру аутентификации и авторизации, после чего создаётся пользовательская сессия, которая, как правило, поддерживается посредством cookies.

Для поддержки работы с cookies в JMeter используется компонент *HTTP Cookie Manager*.

## Установка JMeter

К использованию рекомендуется наиболее актуальная, как правило, последняя версия JMeter (см. п.10.5).

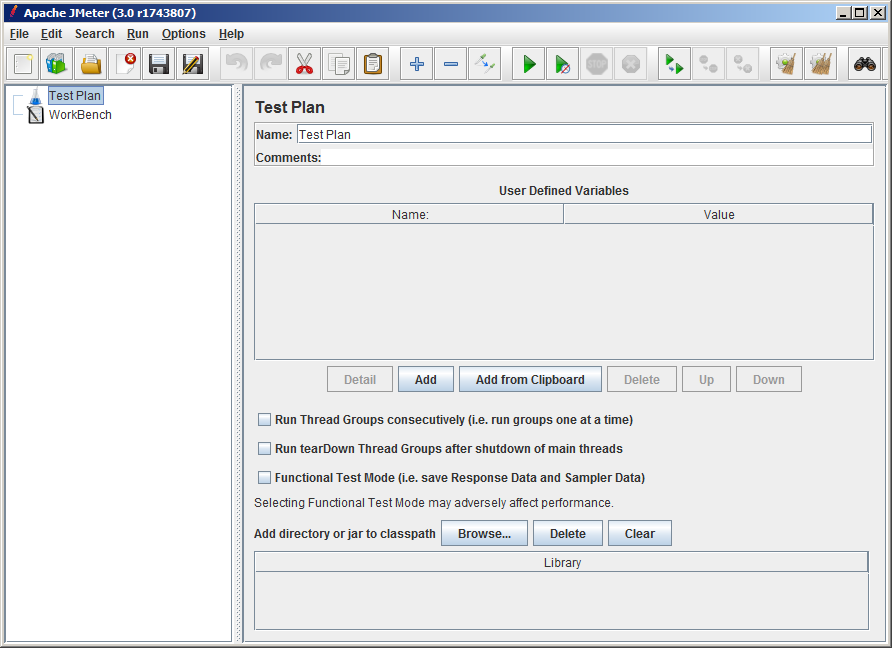
JMeter устанавливается из дистрибутива простым копированием.

## Запуск JMeter

На Windows JMeter запускается из командной строки посредством запуска команды

*%JMETER\_HOME%/bin/jmeter.bat*

После этого должно появиться окно JMeter:



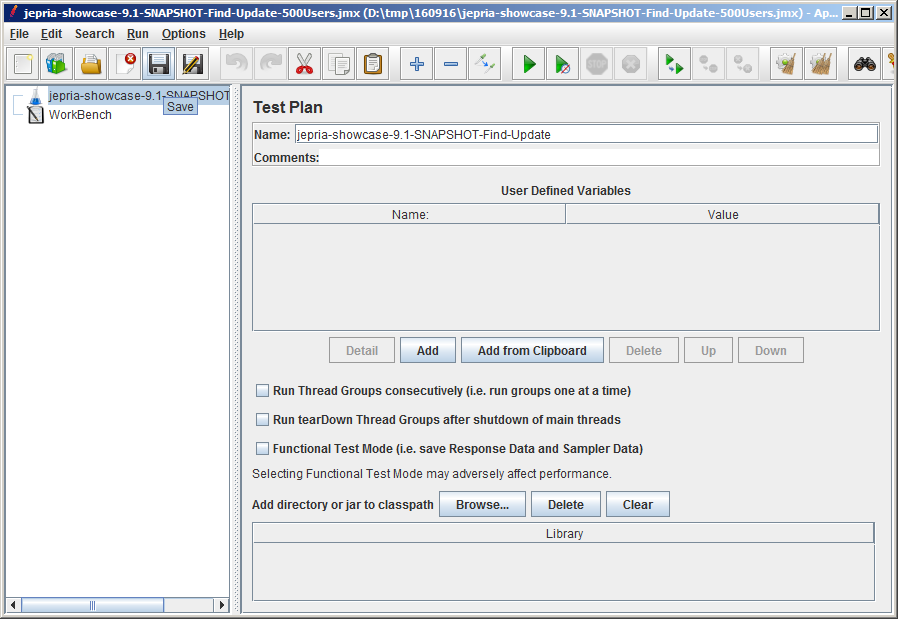
## Создание тест-плана

Типовой тест-план включает в себя следующие составляющие:

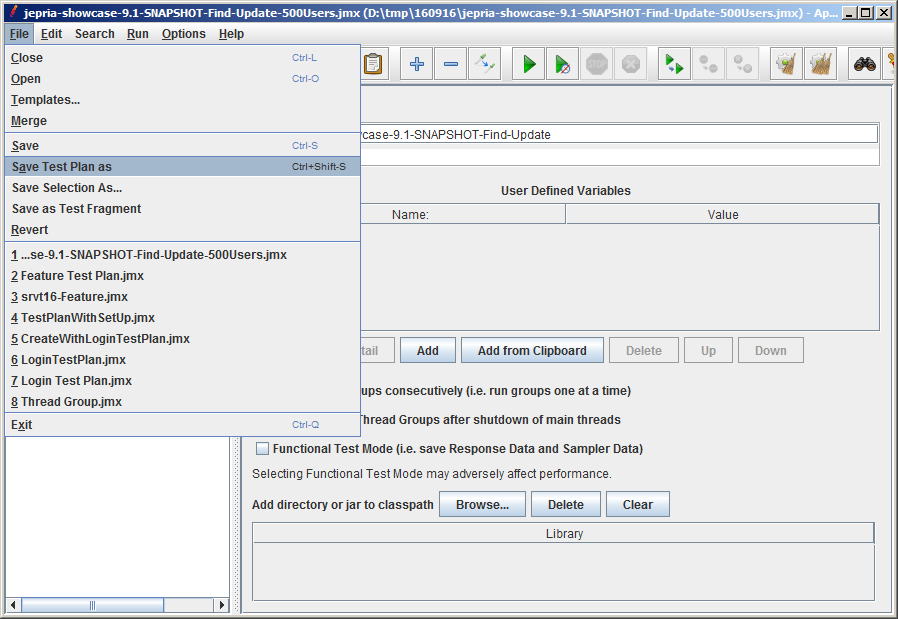
* Модуль поддержки работы с пользовательскими сессиями – *HTTP Cookie Manager*,
* Модули просмотра и анализа результатов работы (*Listeners*)
* Модули тестовых сценариев (*Test Groups*)

Сразу стоит отметить, что, хотя в данной инструкции предлагается определённый рекомендуемый порядок действий, все эти модули можно подключать к тест-плану в любом порядке.

В начале работы тест-плану нужно дать имя, которое говорило бы о главных особенностях создаваемого теста, например, *jepria-showcase-9.1-SNAPSHOT-Find-Update*:

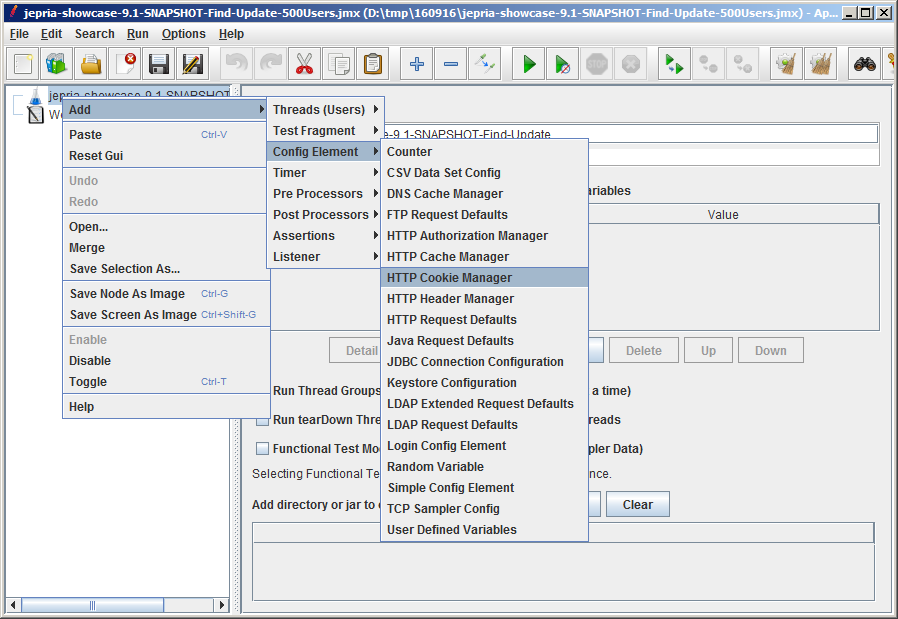


После этого для тест-плана лучше сразу определить место хранения посредством выбора пункта меню *File/Save Test Plan as*:

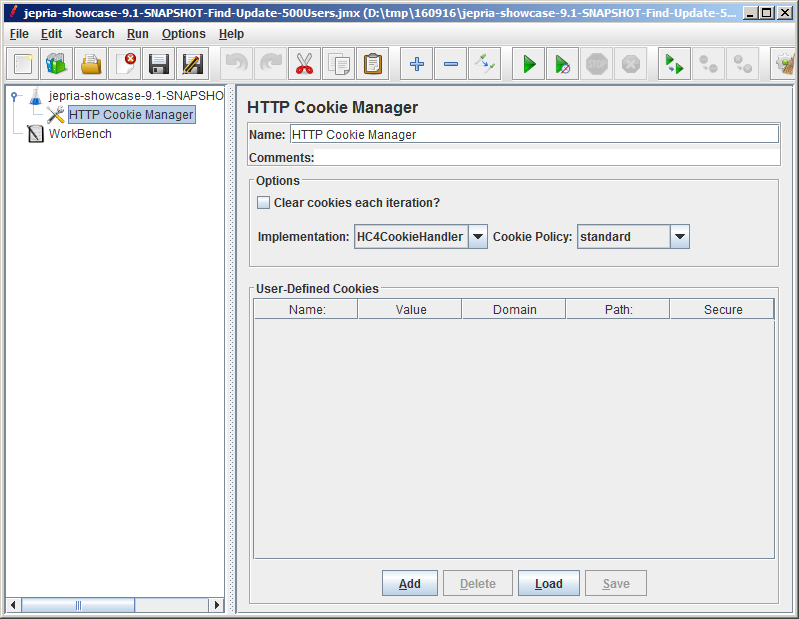


### Подключение модуля поддержки пользовательских сессий - “HTTP Cookie Manager”

Находясь на узле тест-плана, выбираем пункт меню *Add/Config Element/HTTP Cookie Manager*:



Получаем:



Больше никаких действий не требуется, оставляем значения параметров, заданные по умолчанию.

### Подключение модулей просмотра и анализа результатов работы

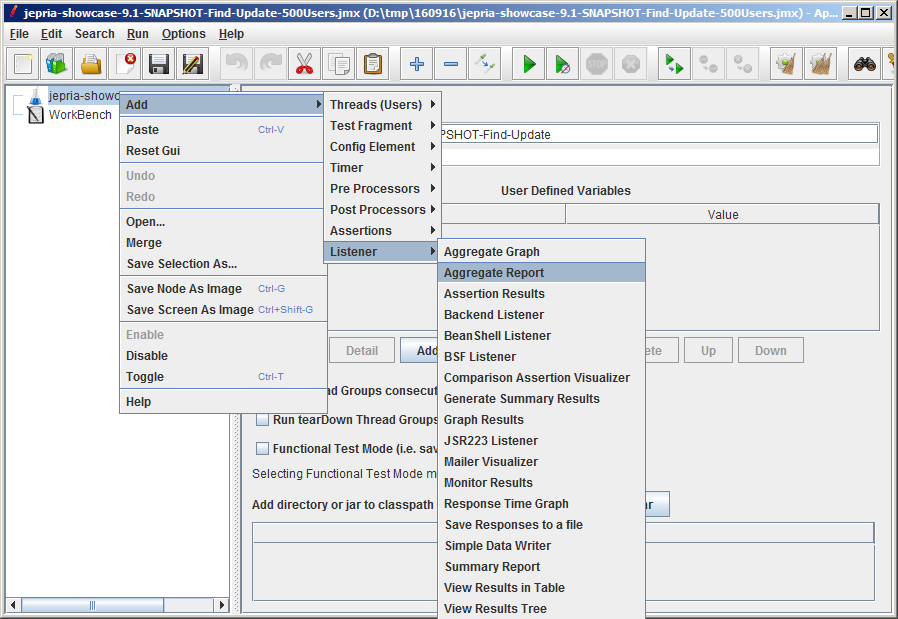
Выбор модулей просмотра результатов выполняется из пункта меню *Add/Listener.*

Из всех модулей необходимыми являются *Aggregate Report* и *Graph Results*.

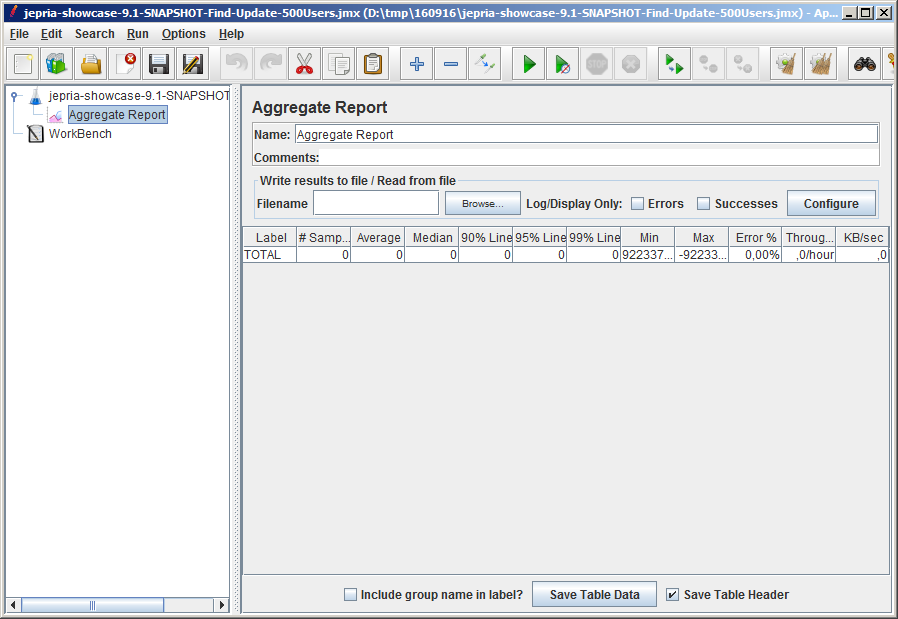
При желании выполнения детального анализа каждого запроса, например, при подозрении на неработоспособность некоторых запросов, можно подключить модули *View Result Tree* и/или *View Results in Table*, но следует учитывать, что эти модули предъявляют повышенные требования к ресурсам оперативной памяти, её нехватка может приводить к *OutOfMemoryException*.

#### Подключение модуля “*Aggregate Report”*

Находясь на узле тест-плана, выбираем пункт меню *Add/Listener/Aggregate Report*:



Получаем:

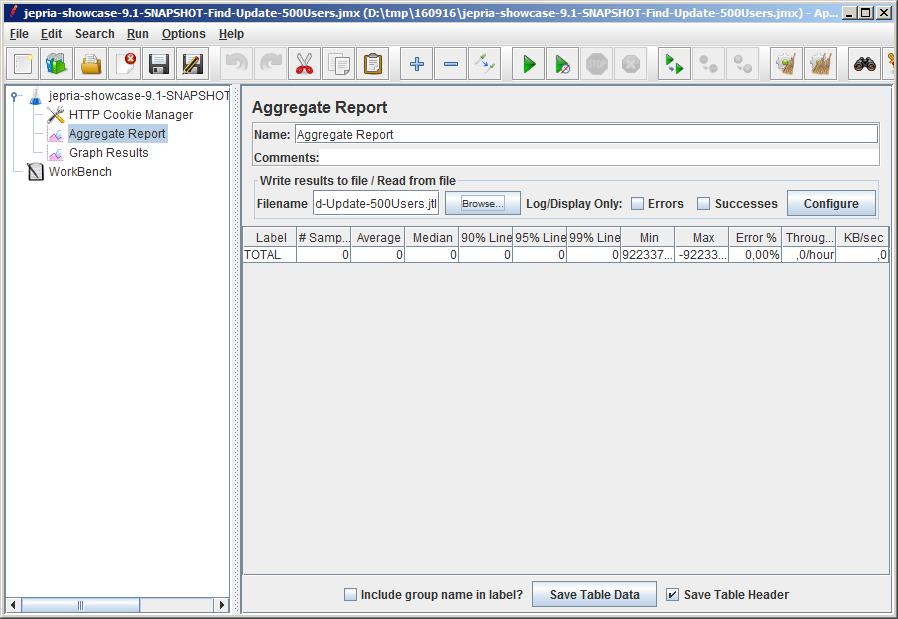


В поле *Filename* нужно указать имя и месторасположение файла, в который будут записываться результаты, и который в дальнейшем можно будет загружать для последующего анализа.

Эти файлы имеют расширение *jtl*. Имя файла рекомендуется создавать на основе имени тест-плана, проще всего – копией имени тест-плана.

В данном случае поле Filename может получить, например, такое значение:

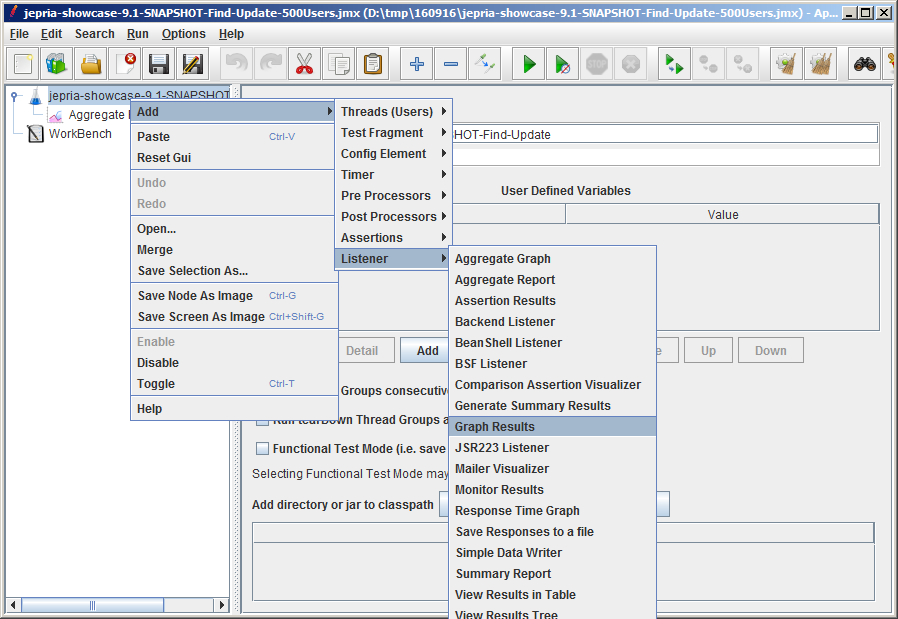
*D:\JepRiaShowcase\Test\jepria-showcase-9.1-SNAPSHOT-Find-Update-500Users.jtl*



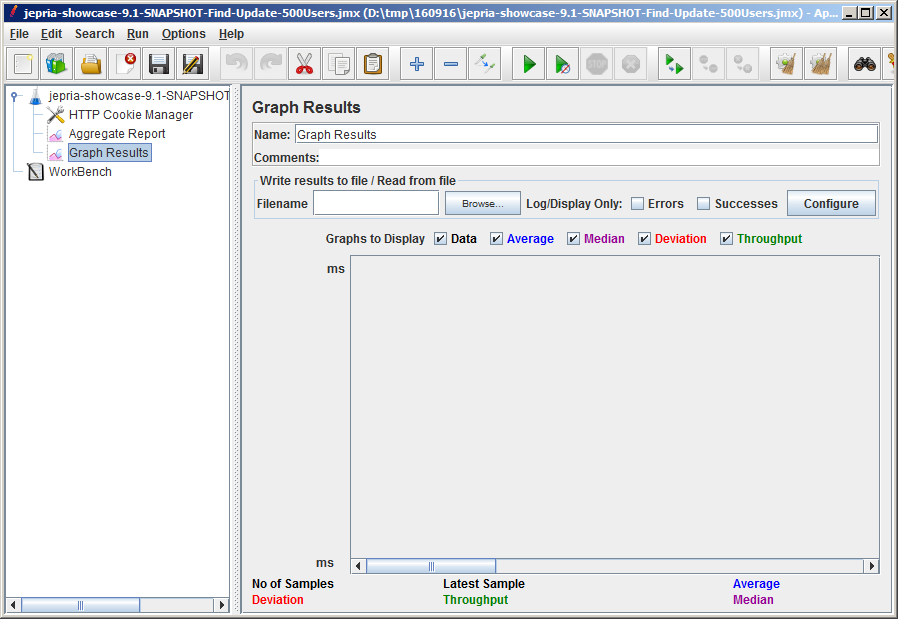
Нужно заметить, что поле *Filename* имеют многие модули из категории Listeners, но заполнять его имеет смысл только в одном из них. Заполнение поля *Filename* в нескольких Listeners приведёт к созданию файлов - дубликатов .

#### Подключение модуля “*Graph Results”*

Находясь на узле тест-плана, выбираем пункт меню *Add/Listener/Graph Results*:



Получаем:



Больше никаких действий не требуется, оставляем значения параметров, заданные по умолчанию.

## Создание узлов тестовых сценариев

Для тестовых сценариев используются модули *Thread Group*. Они бывают трёх типов

* *setup Thread Group* – используется для инициализации теста
* *Thread Group* – используется для выполнения основной части теста
* *tearDown Thread Group* – используется для завершения теста

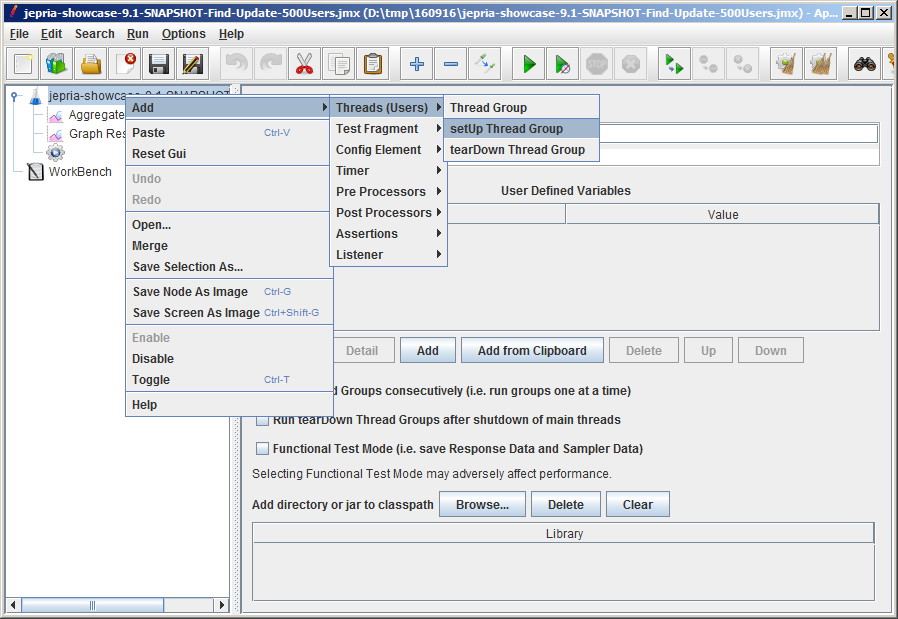
Сценарии инициализации и завершения тестов в данном случае будут использоваться, соответственно, для выполнения входа (Login) и выхода (Logout) из приложения.

Все тестовые сценарии, как таковые, в конечном счёте будут представлены последовательностями HTTP-запросов. Но вначале создаются не сами сценарии, а только узлы с нужными параметрами.

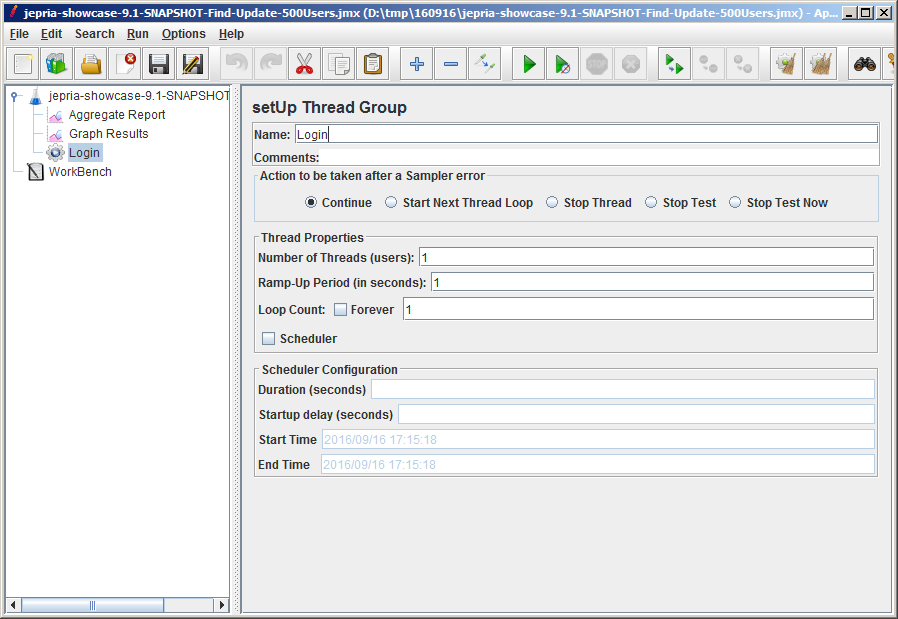
Наполнение узлов запросами будет происходить на следующих этапах подготовки тест-плана.

### Создание узла сценария-инициализации “Login”

Находясь на узле тест-плана, выбираем пункт меню *Add/Threads (Users)/setup Thread Group*:



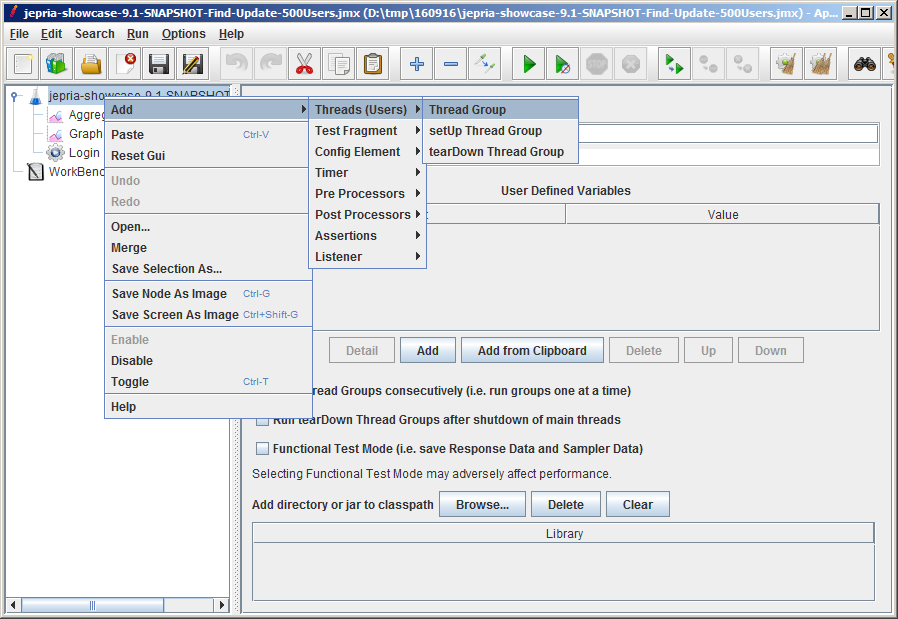
После переименования сохранения получаем:



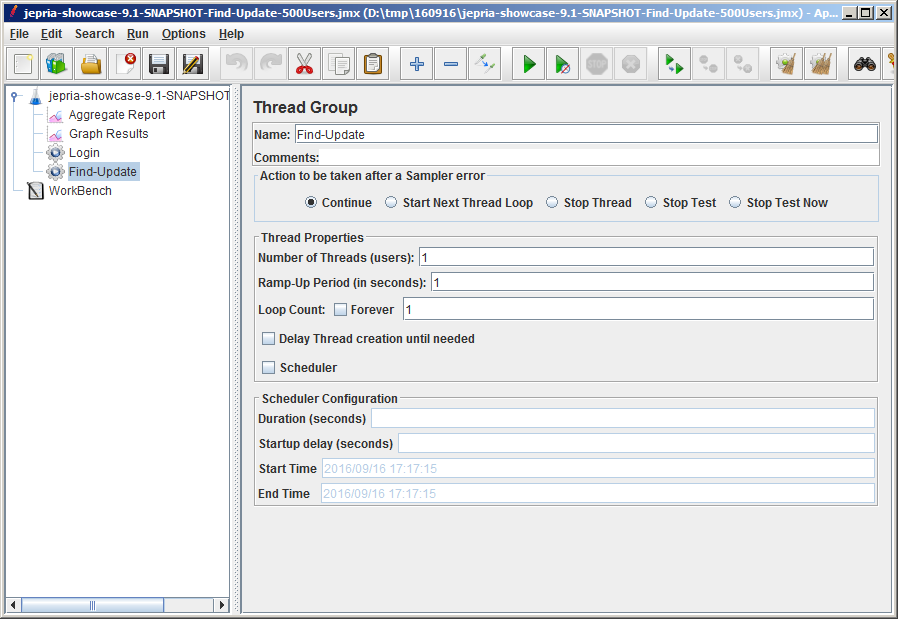
Больше никаких действий на форме не требуется, оставляем значения параметров, заданные по умолчанию.

### Создание узла основого сценария

Находясь на узле тест-плана, выбираем пункт меню *Add/Threads (Users)/Thread Group*:



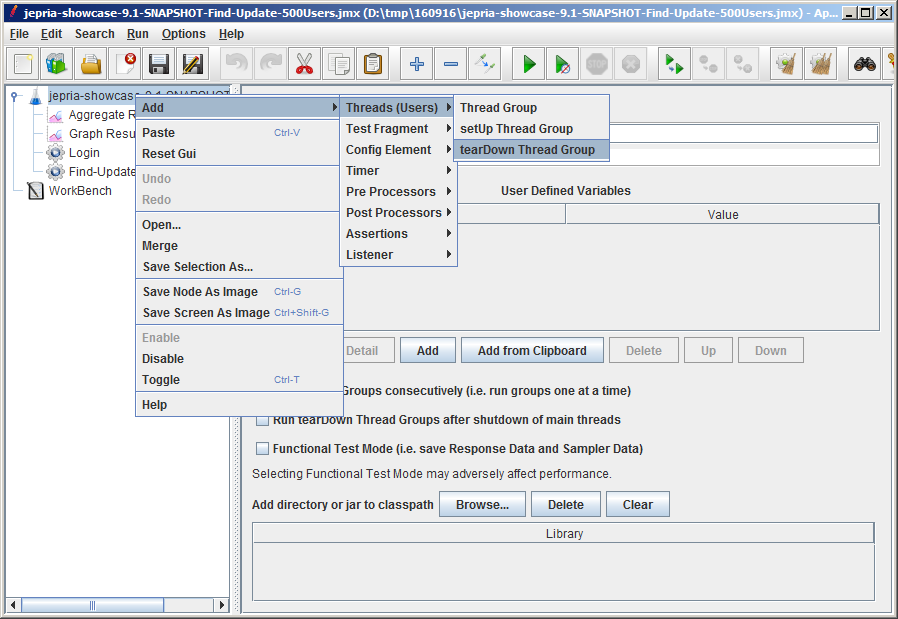
После переименования и сохранения получаем:



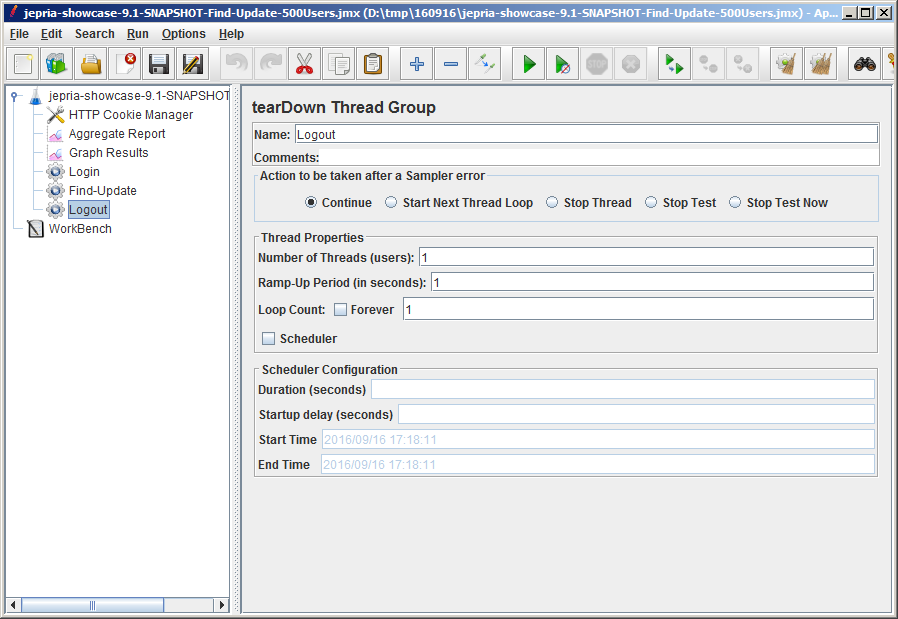
Остальные поля пока не трогаем, вернёмся к ним позже на следующих этапах создания тест-плана.

### Создание узла заключительного сценария “Logout”

Находясь на узле тест-плана, выбираем пункт меню *Add/Threads (Users)/tearDown Thread Group*:



После переименования и сохранения получаем:



Больше никаких действий на форме не требуется, оставляем значения параметров, заданные по умолчанию.

## Запись тестовых сценариев

Самый простой способ создания тестового сценария – автоматическая запись определённых действий пользователя, выполняемых при работе с приложением.

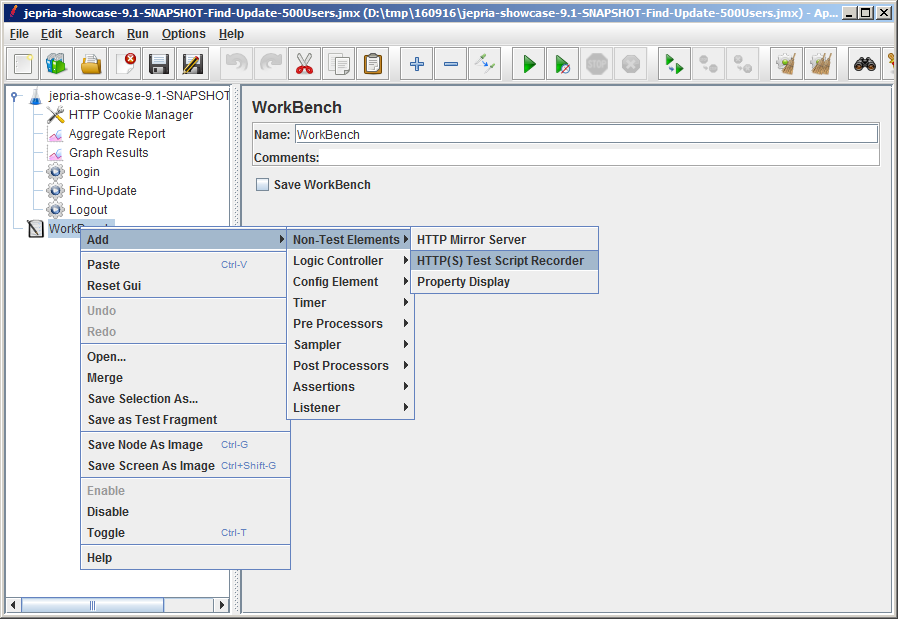
Такая запись выполняется модулем JMeter *HTTP(S) Test Script Recorder*.

Для того, чтобы модуль *HTTP(S) Test Script Recorder* мог выполнять запись сценария, он должен быть указан браузеру в качестве прокси-сервера.

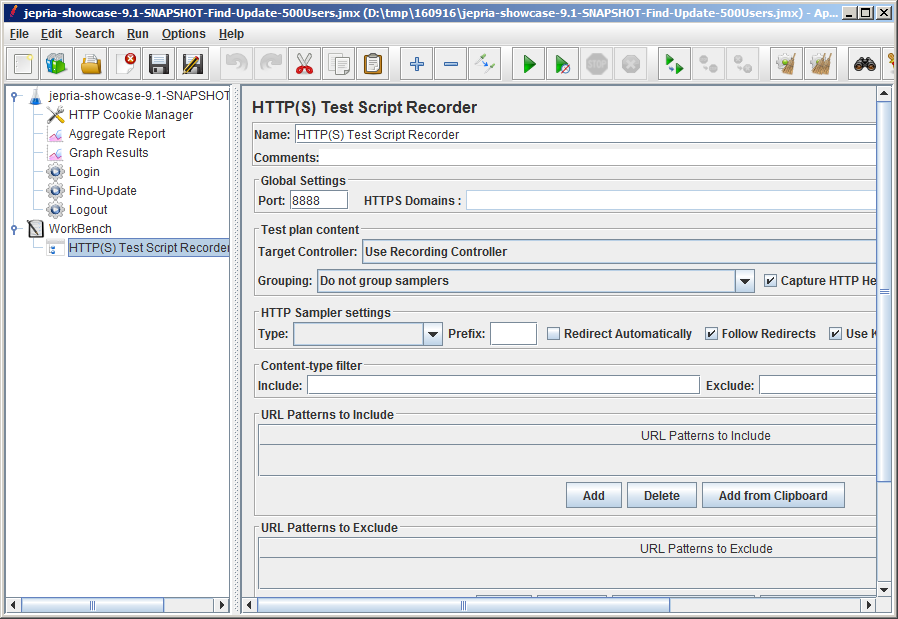
Выступая для браузера в качестве прокси-сервера, модуль *HTTP(S) Test Script Recorder* в то же время выполняет запись запросов, исходящих от браузера в тот узел тестового сценария (Thread Group), который был ему назначен перед записью.

### Подключение модуля “HTTP(S) Test Script Recorder”

Для подключения этого модуля нужно, находясь на узле *WorkBench*, выбрать пункт меню *Add/Non-Test Elements/HTTP(S) Test Script Recorder*:

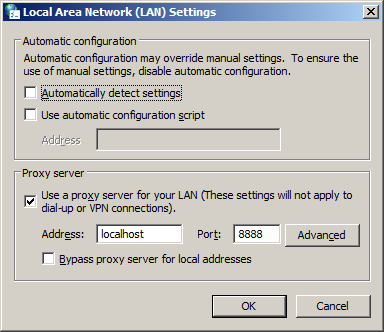


Получаем:



По умолчанию указан порт 8888. Модуль *HTTP(S) Test Script Recorder* будет прослушивать его, выступая для браузера в качестве прокси-сервера. Если порт 8888 занят, модулю необходимо назначить свободный порт.

Каждый браузер имеет свой вид настроек прокси-серверов, например, для Chrome и Internet Explorer настройки будут выглядеть так:

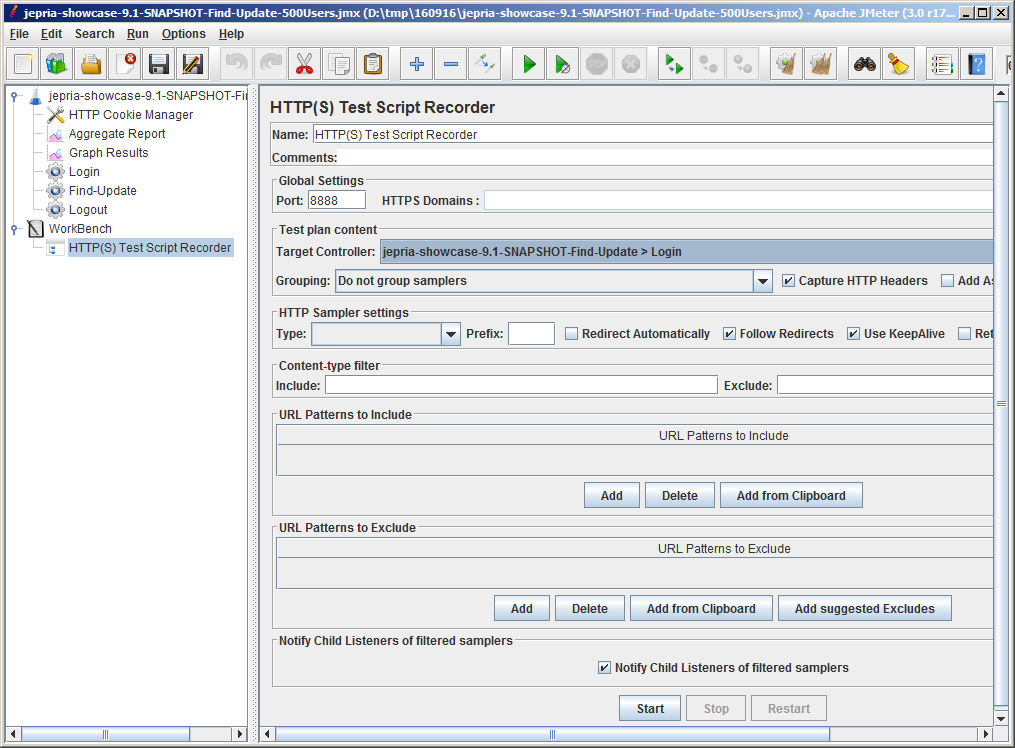


Необходимо обратить внимание на отсутствие галочки в поле *Bypass proxy server for local addresses*.

Необходимо также отметить, что порядок записи сценариев важен. Разумеется, сначала *Login*, затем- основной – в данном случае – *Find-Update*, затем – *Logout*.

### Запись сценария инициализации “Login”

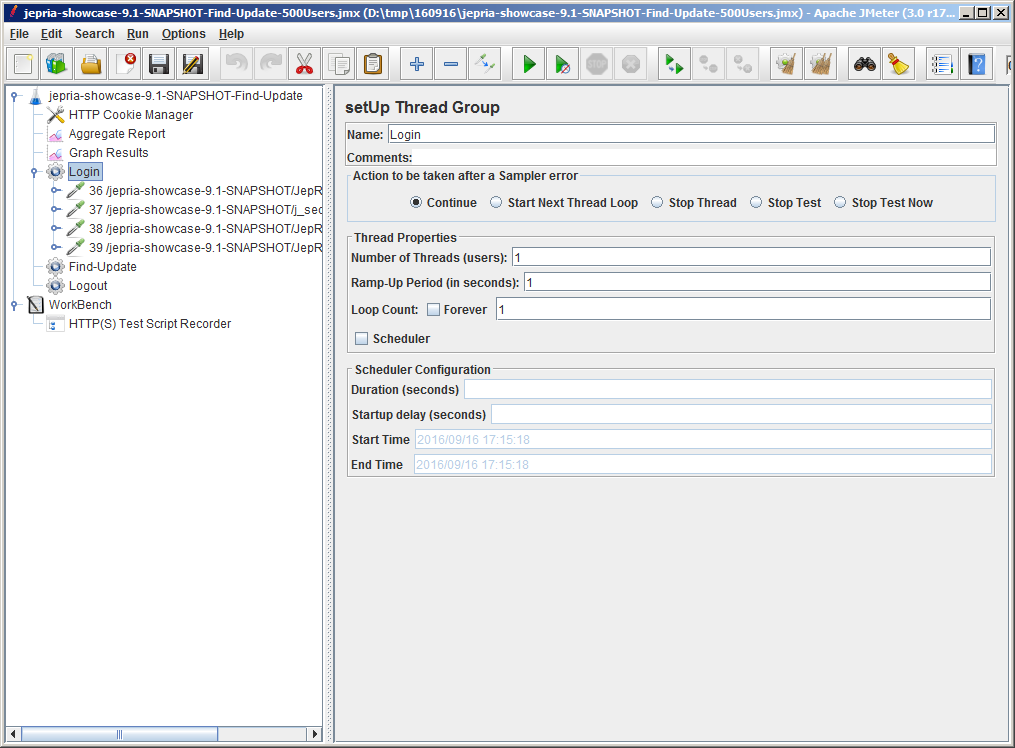
В поле *Target Controller* указывается тестовый сценарий *Login*:



Модуль *HTTP(S) Test Script Recorder* включается в работу нажатием кнопки *Start*.

После включения модуля нужно перейти в окно браузера, выполнить необходимую последовательность действий, вернуться в окно JMeter- в модуль *HTTP(S) Test Script Recorder* и остановить процесс записи нажатием кнопки *Stop*.

После этого узел тестового сценария “Login” должен содержать записанные запросы:



### Запись основного сценария

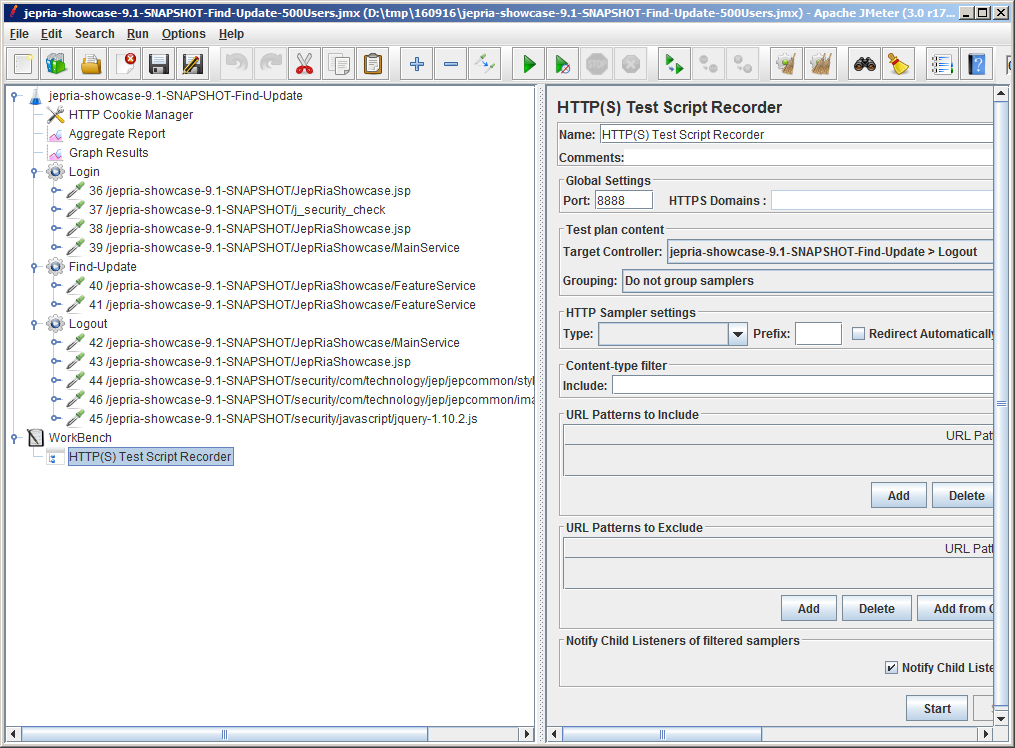
Выполняется аналогично записи сценария *Login* за исключением того, что в поле *Target Controller* указывается тестовый сценарий *Find-Update*.

### Запись сценария “Logout”

Выполняется аналогично записи сценария *Login* за исключением того, что в поле *Target Controller* указывается тестовый сценарий *Logout*.

### Проверка тестовых сценариев

После выполнения записи во все тестовые сценарии, тест план должен иметь такой вид:



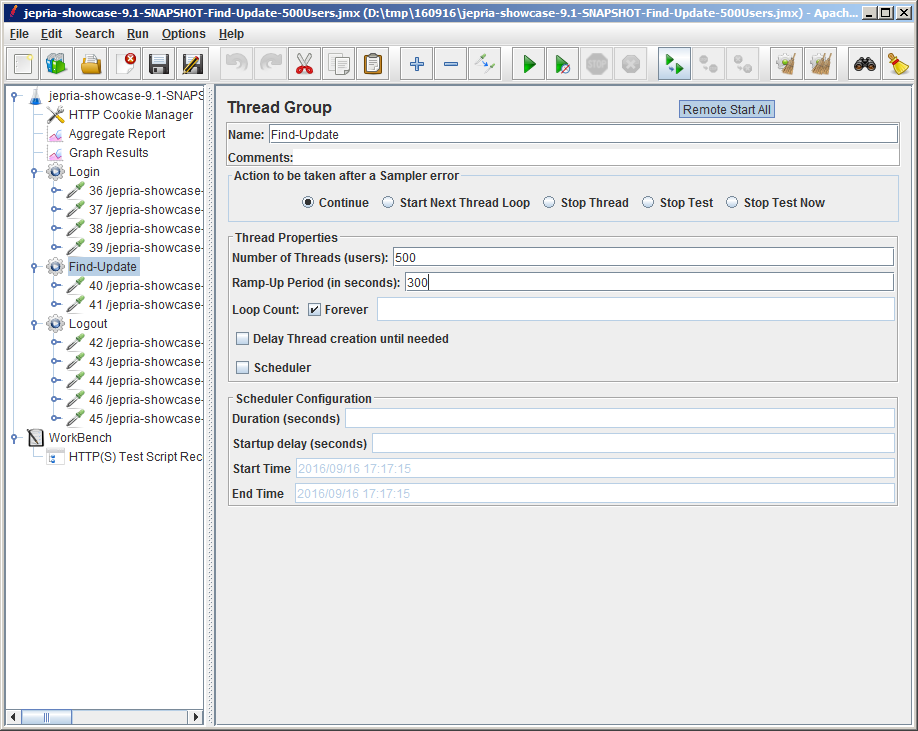
Узлы сценариев должны быть наполнены запросами.

## Настройка основного сценария

Настройка главных параметров режимов выполнения основного сценария делается на узле или под узлом основного сценария. На узле *Find-Update* необходимо задать параметры:

* *Number of Threads (users)* – число пользовательских сессий, в каждой из которых будет выполняться тестовый сценарий
* *Rump-Up period (in seconds)* – время, за которое будут равномерно подключаться новые пользовательские сессии до тех пор, пока их число не достигнет значения *Number of Threads (users)*.
* *Loop Count* – число итераций выполнения сценария.
* Галочка в поле *Forever* означает, что итерации должны выполняться бесконечно. Значение *Loop Count* при этом не имеет смысла и будет стёрто.

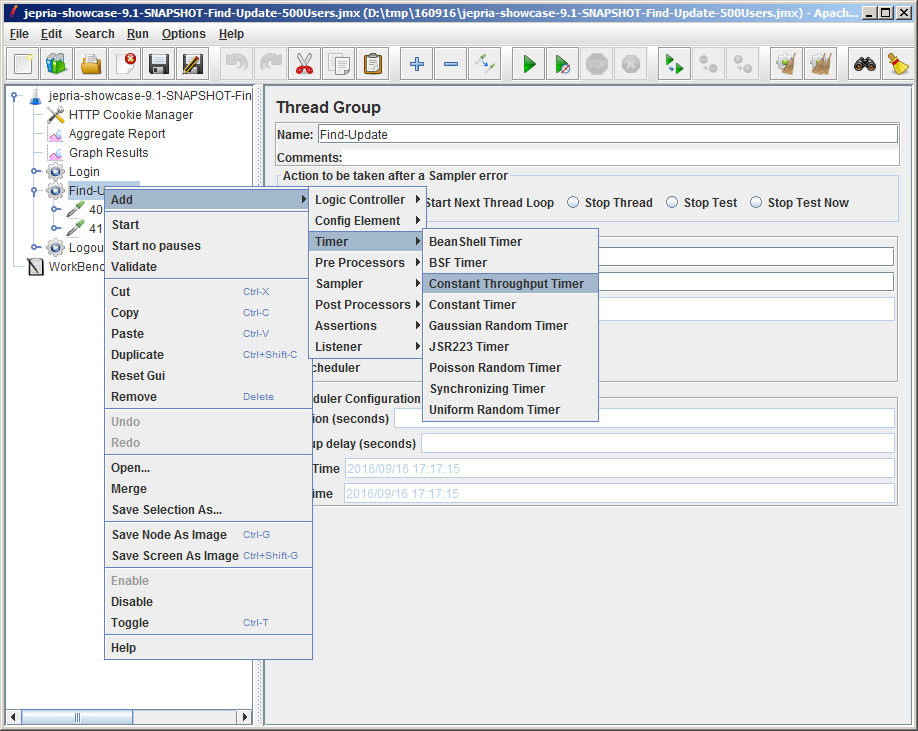
Например:



При такой настройке сценарий *Find-Update* будет выполняться бесконечно, число пользователей будет линейно возрастать от 0 до 500 в течение пяти минут (300 секунд).

### Модуль контроля интенсивности “Constant Throughput Timer”

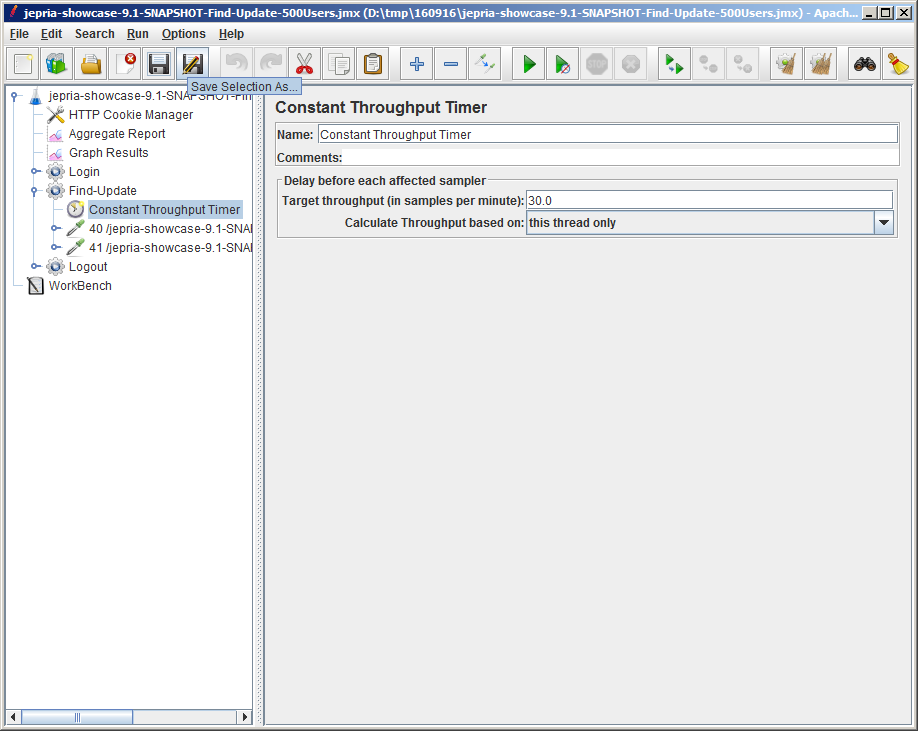
Для управления интенсивностью выполнения запросов основного тестового сценария под его узлом размещается модуль *Constant Throughput Timer*.



Для настроек модуля в данном случае используются два параметра:

* *Target throughput (in samples per minute)* – число запросов, выполняемых за минуту,
* *Calculate Throughput based on* – то, по отношению к чему берётся число запросов (по отношению к каждому «пользовательскому» потоку (thread), по отношению ко всем потокам, и т.д.).

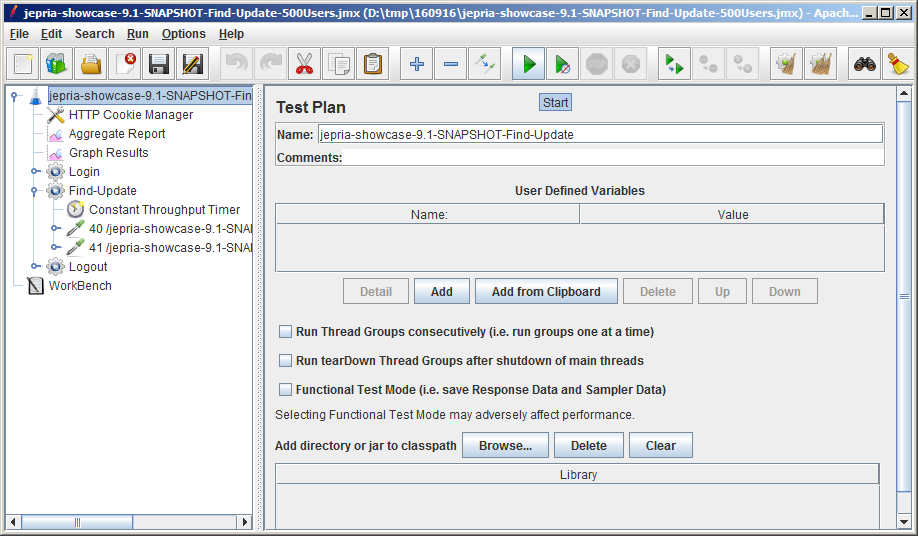
Например, при такой настройке



каждый поток-«пользователь» будет выдавать по запросу в две секунды.

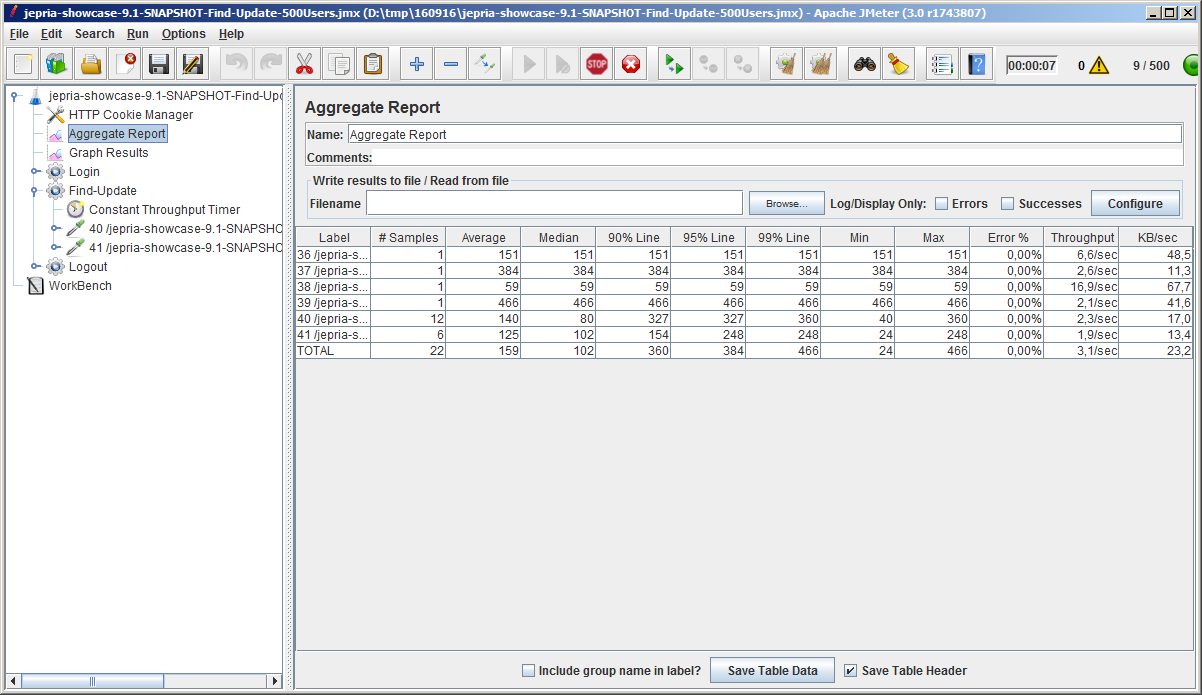
## Запуск теста

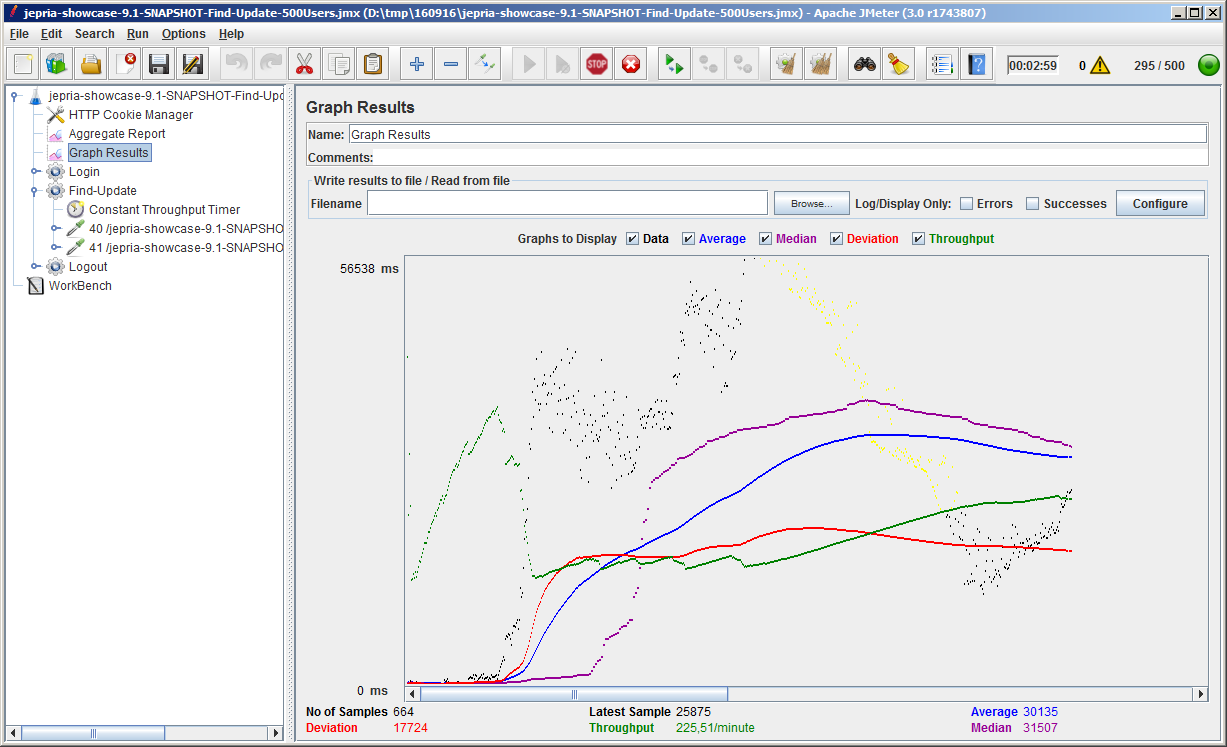
Для запуска теста нужно встать на соответствующий узел и нажать кнопку *Start*.



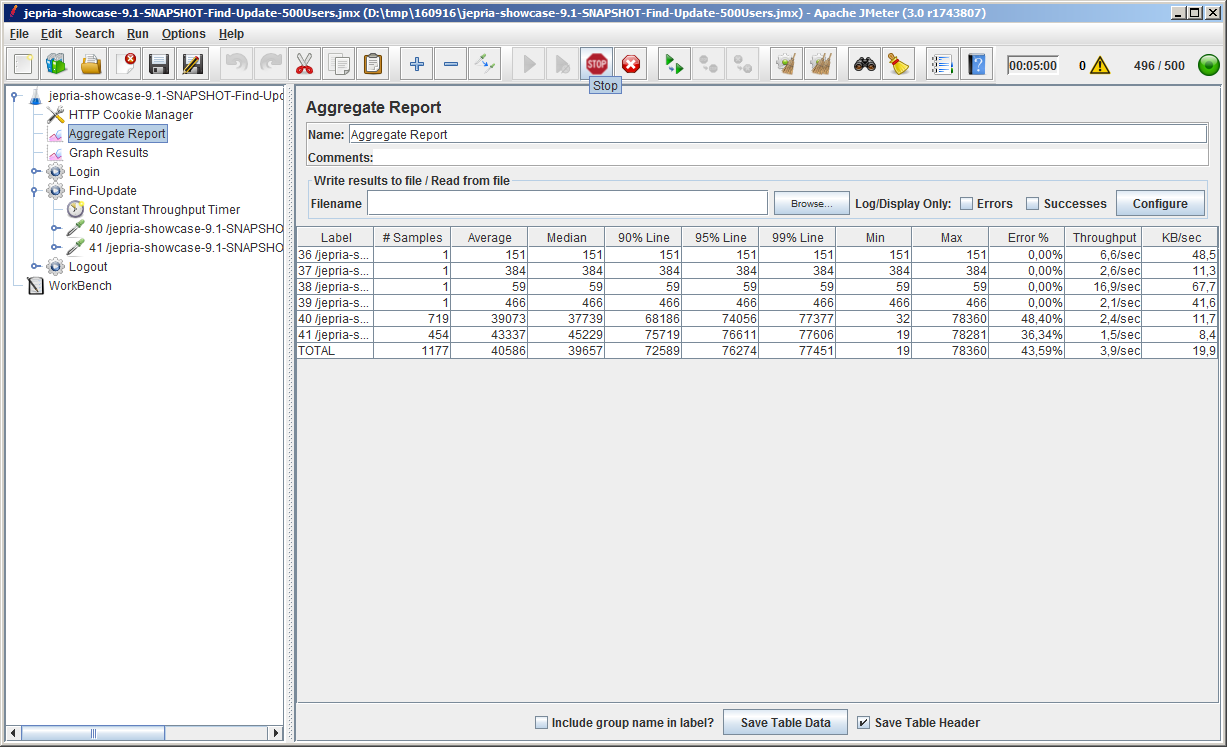
## Просмотр результатов работы теста

JMeter имеет множество модулей Listener-ов, при помощи которых выполняется просмотр и анализ результатов тестов. Например, для просмотра результатов в агрегированном виде используются Listener-ы *Aggregate Report* и *Graph Results*:





Остановка выполнения теста осуществляется нажатием кнопки *STOP*:



Очистка результатов – нажатием кнопок *Clear, ClearAll*.

# Дистрибутивы, библиотеки

В состав дистрибутивов системы входят:

* Дистрибутив *Apache Tomcat*
* Библиотека-коннектор *Tomcat\_JK2*
* Дистрибутив Apache HTTPD
* Коннектор *mod\_jk*

## Дистрибутив Apache Tomcat

[apache-tomcat-7.0.68-windows-x64](http://apache-mirror.rbc.ru/pub/apache/tomcat/tomcat-7/v7.0.68/bin/apache-tomcat-7.0.68-windows-x64.zip)

## Коннектор Tomcat-JK2

Скачивается отсюда: [tomcat-jk2-5.0.28.jar](http://www.java2s.com/Code/JarDownload/tomcat-jk2/tomcat-jk2-5.0.18.jar.zip)

## Дистрибутив Apache HTTPD

[Дистрибутив httpd-2.2.31-x86-r3.zip](http://de.apachehaus.com/downloads/httpd-2.2.31-x86-r3.zip)

## Коннектор MOD\_JK

Скачивается отсюда: [tomcat-connectors-1.2.40-windows-i386-httpd-2.2.x.zip](http://apache-mirror.rbc.ru/pub/apach...s-1.2.40-windows-i386-httpd-2.2.x.zip)

## Дистрибутив Apache JMeter

Скачивается отсюда: [Download Apacht JMeter](http://jmeter.apache.org/download_jmeter.cgi)

# Приложение 1. Настройка HTTPD.

## Настройка кэширования

В приложении приведён перечень общих умолчательных настроек Apache HTTPD.

TODO: Здесь приведено описание для Apache OC4J. Актуализировать

Для корректного кэширования и гарантированного обновления клиентских частей приложений при установке приложений на Tomcat необходимо установить следующие настройки кэширования в конфигурационном файле *httpd.conf* :

После правила запрета доступа к файлам, начинающихся с .ht:

*<Files ~ "^\.ht">*

*Order allow,deny*

*Deny from all*

*</Files>*

Добавить правила:

*#*

*# Accurate caching of JepRia applications.*

*#*

*<Files \*.nocache.\*>*

*ExpiresActive on*

*ExpiresDefault "now"*

*Header* ***merge*** *Cache-Control "public, max-age=0, must-revalidate"*

*</Files>*

*<Files \*.cache.\*>*

*ExpiresActive on*

*ExpiresDefault "now plus 1 year"*

*</Files>*

Полезные ссылки по настройке кэширования для GWT-приложений:

<http://www.gwtproject.org/doc/latest/DevGuideCompilingAndDebugging.html#perfect_caching>

<http://seewah.blogspot.ru/2009/02/gwt-tips-2-nocachejs-getting-cached-in.html>

## Настройки, связанные с безопасностью сервера приложений

### Скрытие названия и версии сервера приложений

TODO: Здесь приведено описание для Apache OC4J. Актуализировать.

Для того, чтобы скрыть версию и наименование сервера приложения, необходимо выполнить следующие действия в конфигурационном файле httpd.conf:

* 1. **Загрузить модуль mod\_security:**

В секции Dynamic Shared Object (DSO) Support прописать:

LoadModule security\_module  modules/ApacheModuleSecurity.dll

В секции добавления модулей (ClearModuleList):

AddModule mod\_security.c

* 1. **Установить значение опции ServerSignature:**

ServerSignature Off

* 1. **Установить значение опции SecServerSignature**

SecServerSignature “Unknown”

*Примечание 1*: Вместо “Unknown” может быть любое не пустое значение – фактически данное значение будет возвращаться в заголовке ответа “Server”

*Примечание 2:*  В том случае, если значение SecurityTokens по какой-то причине перекрыто, необходимо установить его в Full

# Приложение 2.Пример настройки файла *server.xml* в Tomcat.

Ниже приводится пример файла server.xml Tomcat, настроенного на взаимодействие с Apache HTTPD. Изменённые/добавленные фрагменты выделены ***жирным шрифтом (изменение по кодировке и SSO)***:

*<?xml version='1.0' encoding='utf-8'?>*

*<Server port="8105" shutdown="SHUTDOWN">*

*…*

*<Connector port="8180"* ***URIEncoding="UTF-8****" protocol="HTTP/1.1"*

*connectionTimeout="20000"*

*redirectPort="8443" />*

*…*

*<Engine name="Catalina" defaultHost="localhost">*

*<Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">*

*…*

***<Valve className="org.apache.catalina.authenticator.SingleSignOn" />***

*…*

*</Host>*

*</Engine>*

*</Service>*

*</Server>*

# Приложение 3.Пример настройки файла mod\_jk.conf

Пример настройки приводится для обеспечения запуска примеров Tomcat:

*########## Modified Auto generated on Wed Mar 30 19:21:54 MSK 2016##########*

*<IfModule !mod\_jk.c>*

*LoadModule jk\_module "modules/mod\_jk.so"*

*</IfModule>*

*JkWorkersFile "conf/jk/workers.properties"*

*JkLogFile "logs/mod\_jk.log"*

*JkLogLevel emerg*

*# Select the log format*

*JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"*

*# JkOptions indicate to send SSL KEY SIZE,*

*JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories*

*# JkRequestLogFormat set the request format*

*JkRequestLogFormat "%w %V %T"*

*# Send everything for context /ws to worker* ***worker1***

*JkMount /ws* ***worker1***

*JkMount /ws/\** ***worker1***

*# Send everything for context /examples to worker* ***worker1***

*JkMount /examples* ***worker1***

*JkMount /examples/\** ***worker1***

*JkMount /jepriashowcase-1-1* ***worker1***

*JkMount /jepriashowcase-1-1/\** ***worker1***

*JkMount /jepriashowcase-1-2* ***worker1***

*JkMount /jepriashowcase-1-2/\** ***worker1***

*JkMount /jepriashowcase-2-1* ***worker2***

*JkMount /jepriashowcase-2-1/\** ***worker2***

*JkMount /jepriashowcase-2-2* ***worker2***

*JkMount /jepriashowcase-2-2/\** ***worker2***

Замечание:

Имена worker-ов (в данном случае - ***worker1****,* ***worker2***) должны соответствовать именам, определённым в файле *workers.properties*.

# Приложение 4



Рисунок 2. Снижение нагрузки с Tomcat посредством переноса ответственности за статические ресурсы на HTTPD.

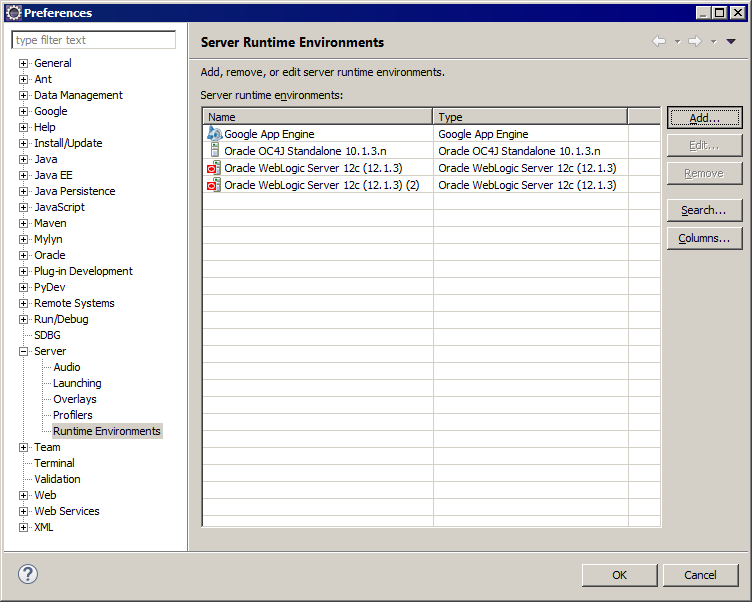


Рисунок 3.Схема балансировки нагрузки.

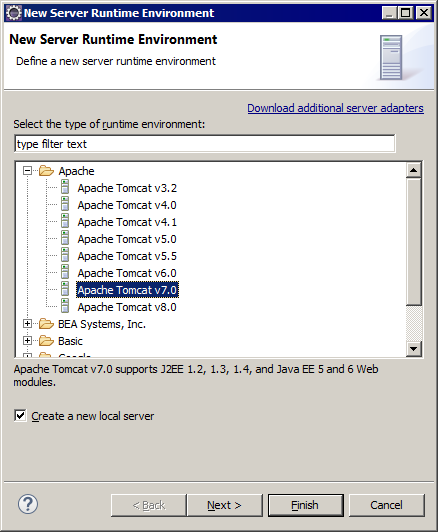
# Приложение 5. Конфигурирование Tomcat для отладки в Eclipse

На машине разработчика удобнее всего работать не с сервисом Tomcat, а с процессом, стартуемым из среды разработки Eclipse. В этом случае в консоли отображается лог сервера и ошибки; кроме того, становятся доступными возможности серверной отладки. Для этого требуется выполнить следующие действия.

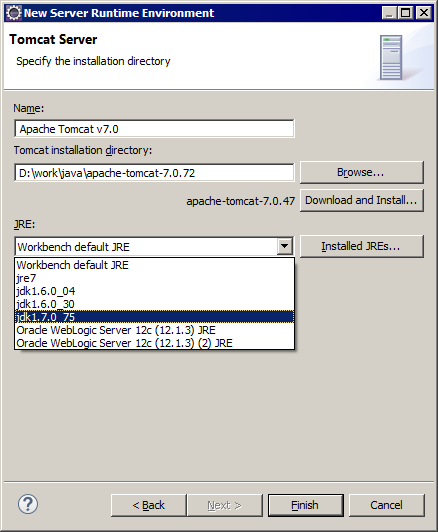
1. Предварительно необходимо установить и сконфигурировать Tomcat в соответствии с указаниями в разделах 4, 6, 7.1,7.2, 8.1.
2. Далее нужно конфигурировать серверное окружение (server runtime) Tomcat и создать сервер. Для этого в строке меню выбрать *Window -> Preferences* и в дереве слева выбрать раздел *Server -> Runtime Environments*. Нажать кнопку *Add*…



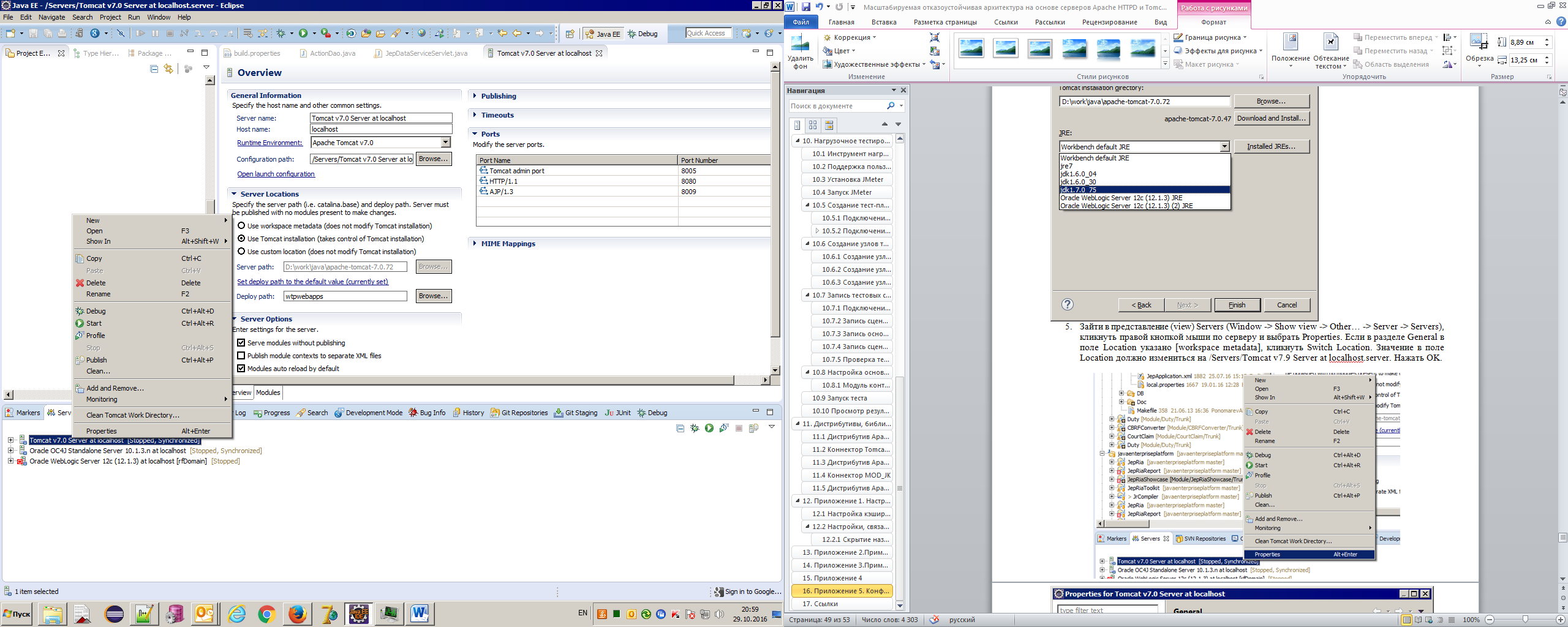
1. В появившемся окне выбрать *Apache Tomcat v7.0*. Установить флаг *Create a new local server*. Нажать *Next*.

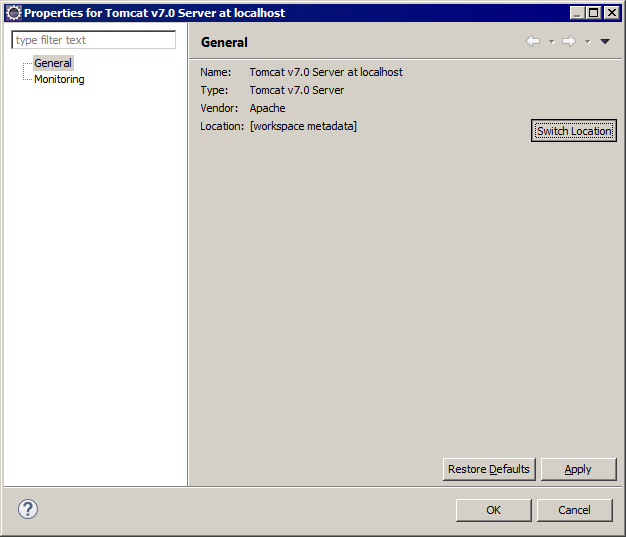


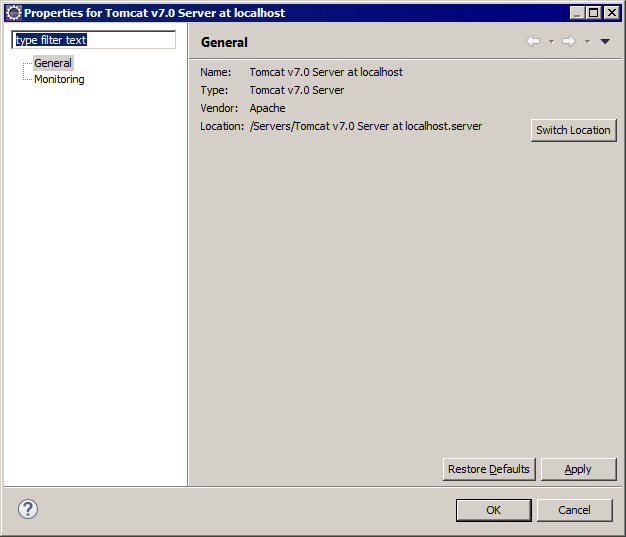
1. Ввести имя сервера (по умолчанию Apache Tomcat v7.0, можно оставить это название). В поле *Tomcat installation directory* ввести путь установки Tomcat, в комбобоксе *JRE* выбрать используемую версию jdk. Нажать *Finish*.



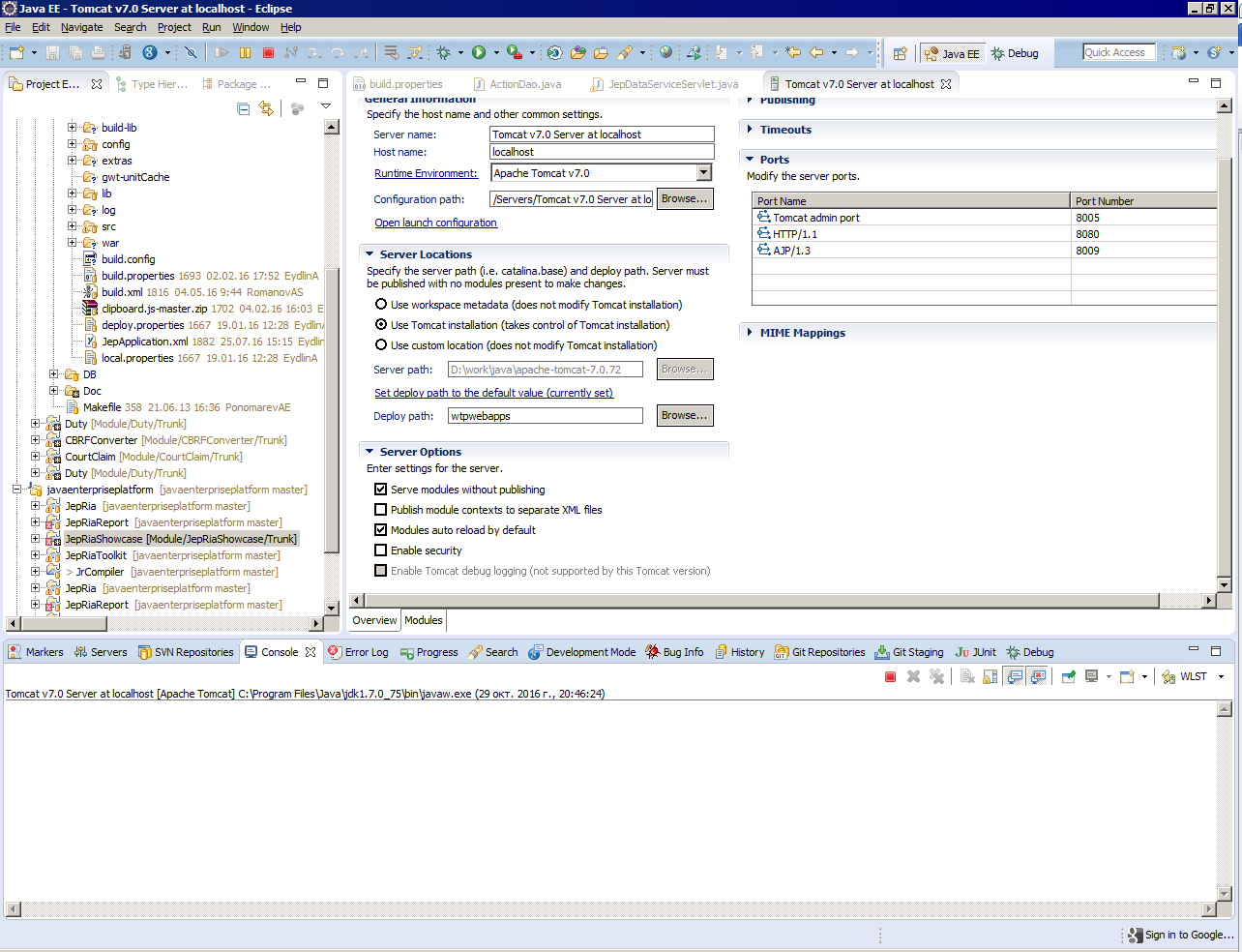
1. Зайти в представление (view) *Servers* (*Window -> Show view -> Other… -> Server -> Servers*), кликнуть правой кнопкой мыши по серверу и выбрать *Properties*. Если в разделе General в поле *Location* указано *[workspace metadata]*, кликнуть *Switch Location*. Значение в поле *Location* должно измениться на */Servers/Tomcat v7.0 Server at localhost.server*. Нажать *OK*. Данное изменение необходимо, т.к. по умолчанию Eclipse запускает Tomcat в рабочем пространстве, что предполагает установку приложений средствами Eclipse (с помощью Publish), а не с помощью ant.



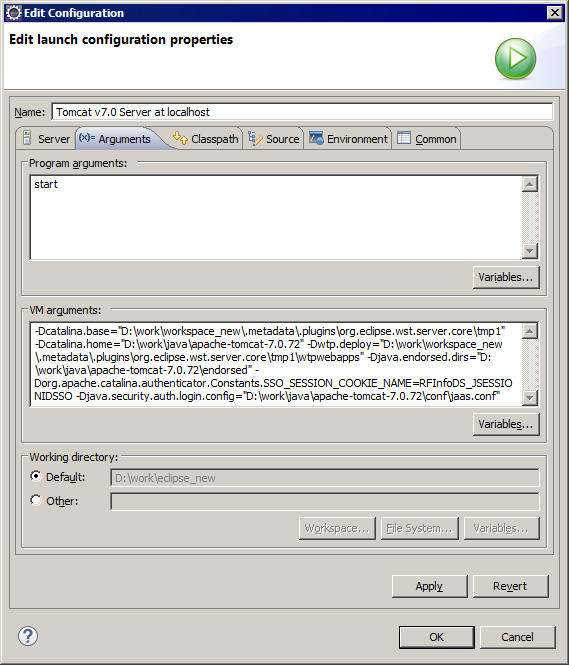




1. В представлении *Servers* дважды кликнуть по созданному серверу. Откроется окно конфигурации. В разделе *Server Locations* выбрать *Use Tomcat installation (takes control of Tomcat installation)*, что также необходимо при ручной установке приложений. В разделе *Server options* установить флаг *Serve Modules without publishing*.



1. Нажать *Open launch configuration* – откроется окно *Edit configuration*. Зайти во вкладку *Arguments* и дополнить поле *VM arguments* аргументами, необходимыми для работы SSO, как это было сделано в файле service.bat в разделе 7.3. **Важно:** 1) вместо %CATALINA\_BASE% требуеся указать полный путь до файла jaas.conf; 2) в отличие от service.bat, разделителями аргументов командной строки служит не точка с запятой, а пробел.



Нажать *OK* и сохранить конфигурацию.

1. Сервер готов к запуску. В представлении *Servers* кликнуть правой кнопкой мыши и выбрать *Debug*. Убедиться в корректной работе, зайдя по URL <http://localhost:8080/manager/> и попробовав установить с помощью ant приложение.
2. Если в процессе запуска Tomcat возникают исключения вида org.xml.sax.SAXParseException, то на вкладке *Arguments* из п.7 добавить аргумент *-Djavax.xml.parsers.SAXParserFactory= com.sun.org.apache.xerces.internal.jaxp.SAXParserFactoryImpl*.

# Ссылки

1. [Apache HTTPD, mod\_jk, Apache Tomcat, Linux](http://programador.ru/apache-mod_jk-tomcat/)
2. [Using Apache HTTP Server on Microsoft Windows](http://httpd.apache.org/docs/current/platform/windows.html#down)
3. [Installing Tomcat 6 with Java 6 and Apache 2.2 on Windows Server 2003 x64](http://www.anchor.com.au/hosting/dedicated/tomcat_with_apache_on_windows_x64)
4. Полезный ресурс: [Installing Apache and Tomcat and Mysql on Windows 7](http://boldinventions.com/index.php?option=com_content&view=article&id=95%3Ainstalling-apache-and-tomcat-and-mysql-on-window-7&catid=34%3Acategory-electronics-articles&Itemid=53)
5. Помогло при настройке *mod\_jk.conf*: [How To Configure Tomcat to work with Apache](http://www.ntu.edu.sg/home/ehchua/programming/howto/apacheplustomcat_howto.html)
6. [Apache JMeter](http://jmeter.apache.org/)
7. [Professional Apache Tomcat 6](https://www.amazon.com/Professional-Apache-Tomcat-Vivek-Chopra/dp/0471753610)
8. [Apache Commons Daemon](https://commons.apache.org/proper/commons-daemon/index.html)