# **Инструкция по сборке JepRia приложения средствами Maven-a (на примере приложения JepRiaShowcase).**

Обновлённая структура приложения на фреймворке JepRia предполагает наличие встроенных (дочерних) модулей, собранных при помощи различных технологий сборки. Инструкция посвящена особенностям сборки JepRia приложения средствами Maven-а. Алгоритм сборки рассматривается на примере демонстрационного приложения JepRiaShowcase, состоящего из 2х дочерних модулей, один из которых собирается средствами Ant-а, а другой – средствами Maven-а. Maven осуществляет сборку проекта на основе специального файла с описанием проекта pom.xml (декларативный подход). Указанные файлы pom.xml были разработаны для одного из дочерних модулей приложения JepRiaShowcase (service-rest) и для всего приложения JepRiaShowcase. Может потребоваться только их незначительная доработка (например, для изменения настроек развёртывания приложения на сервер приложений Tomcat). Также в инструкции рассматриваются возможные варианты настройки Maven-а для корректной работы в рамках корпоративной сетевой инфраструктуры.

## **Установка и конфигурирование Maven-а.**

Установку и настройку Maven-а предлагается провести в соответствии со следующей последовательностью действий:

* 1. Скачать [архив](https://maven.apache.org/download.cgi) с Maven-ом и распаковать в любую папку, например C:\tools\maven.
  2. Задать системную переменную среды M2\_HOME со значением равным пути установки Maven-а, то есть M2\_HOME = C:\tools\maven.
  3. Добавить в системную переменную среды Path путь %M2\_HOME%\bin. После этого Maven должен запускаться из командной строки командой mvn.
  4. Скопировать конфигурационный файл C:\tools\maven\conf\settings.xml в папку C:\Users\%UserName%\.m2 (при необходимости создать указанную папку из командной строки командой mkdir .m2).
  5. Настроить параметры подключения Maven-а к центральному репозиторию. Указанная настройка будет зависеть от конкретной ситуации. Основная проблема – Maven некорректно работает через прокси сервер с NTLM аутентификацией (в нашей корпоративной сети используется именно такой). Соответственно, если сборка осуществляется с компьютера за пределами корпоративной сети, то, скорее всего, дополнительных настроек подключения не потребуется. В этом случае Maven попытается скачать зависимости из центрального репозитория напрямую. Если же сборка осуществляется с компьютера в корпоративной сети, то предлагается выбрать любой из следующих вариантов настройки:
     + 1. Задать корпоративное зеркало для центрального репозитория Maven-а, добавив в конфигурационный файл .m2\settings.xml (из профайла пользователя) в секцию <mirrors> следующий элемент:  
            
          <mirror>  
           <id><произвольный уникальный идентификатор зеркала></id>  
           <mirrorOf>central</mirrorOf>  
           <url><путь до корпоративного репозитория Nexus></url>  
          </mirror>
       2. Установить и настроить дополнительный локальный прокси сервер CNTLM, который позволит Maven-у корректно подключиться к центральному репозиторию через корпоративный прокси сервер. Инструкция по настройке прокси сервера CNTLM может быть найдена по следующему адресу:  
          <https://github.com/Jepria/doc/blob/master/cntlm-settings-for-npm-yarn-maven.md>
  6. Подключить в качестве дополнительного репозитория для поиска зависимостей и плагинов локальную папку с бинарным репозиторием, добавив в конфигурационный файл .m2\settings.xml (из профайла пользователя) в секцию <profiles> следующий элемент:  
       
      <profile>  
       
      <repositories>  
      <repository>  
      <id>binhome-local-deploy</id>  
      <name>binhome-local-deploy-repo</name>  
      <url>file:${BIN\_HOME}\deploy</url>  
      </repository>  
      <repository>  
      <id>binhome-local-build</id>  
      <name>binhome-local-build-repo</name>  
      <url>file:${BIN\_HOME}\build</url>  
      </repository>  
      </repositories>  
       
      <pluginRepositories>  
      <pluginRepository>  
      <id>local-maven-plugins</id>  
      <name>local-maven-plugins-repo</name>  
      <url>file:${BIN\_HOME}\build</url>  
      </pluginRepository>  
      </pluginRepositories>  
       
      <id>binhome</id>  
      </profile>  
       
     Тут под бинарным репозиторием понимается папка, синхронизированная с GitHub репозиторием:  
     <https://github.com/Jepria/bin-repo>  
     На момент написания инструкции была доступна копия бинарного репозитория на SourceForge по следующему адресу:  
     <http://svn.code.sf.net/p/javaenterpriseplatform/svn/Bin>  
     При сборке приложения с компьютера в корпоративной сети также доступно зеркало бинарного репозитория:  
     svn://srvsvn/JEP/Bin  
     Системная переменная среды ${BIN\_HOME} должна ссылаться на указанный бинарный репозиторий, например ${BIN\_HOME} = c:\SVN\JEP\Bin\.
  7. Для активации описанного в предыдущем пункте профиля добавить в тот же конфигурационный файл settings.xml в секцию <settings> следующий элемент:  
       
      <activeProfiles>  
      <activeProfile>binhome</activeProfile>  
      </activeProfiles>  
       
     Пример валидного конфигурационного файла settings.xml:  
     
  8. В качестве альтернативы применения бинарного репозитория все необходимые зависимости могут быть скопированы непосредственно в кэш Maven-а (в папку .m2) по валидному пути.

## **Сборка приложения JepRiaShowcase средствами Maven-a.**

Для сборки приложения JepRiaShowcase средствами Maven-а предлагается следующая последовательность действий:

* 1. Обновить бинарный репозиторий. Убедиться, что бинарный репозиторий содержит следующие Maven плагины:  
       
     snapshotupdate-maven-plugin-<some version>.jar  
     collisioncheck-maven-plugin-<some version>.jar  
     cleanall-maven-plugin-<some version>.jar
  2. Скачать/обновить приложение JepRiaShowcase (исходники) из ветки master репозитория  
     <https://github.com/Jepria/jepria-showcase>  
     На момент написания инструкции также была доступна копия приложения JepRiaShowcase на SourceForge (SVN, ветка TRUNK):  
     <https://svn.code.sf.net/p/javaenterpriseplatform/svn/Module/JepRiaShowcase/Trunk>
  3. Убедиться в наличии файла pom.xml в папке App приложения JepRiaShowcase. На текущий момент набор дочерних модулей, включаемых в итоговый war-архив приложения JepRiaShowcase, задаётся явно в файле pom.xml (это модули gwt и service-rest). Поэтому может потребоваться незначительная доработка файла pom.xml при добавлении нового дочернего модуля в приложение JepRiaShowcase. Потребуется настроить плагины (по аналогии с имеющейся настройкой для модулей gwt и service-rest в элементе <build><plugins> файла pom.xml) maven-war-plugin, exec-maven-plugin, truezip-maven-plugin и collisioncheck-maven-plugin.
  4. Открыть консоль, перейти в папку App приложения JepRiaShowcase, убедиться в наличии в папке App файла pom.xml и запустить сборку приложения одной из следующих команд:
* mvn package – сборка war-файла приложения (будет скопирован в папку lib).
* mvn clean package – сборка war-файла приложения с предварительной поверхностной очисткой результатов предыдущей сборки (очищаются только папки App\target и App\lib).
* mvn clean-all package – сборка war-файла приложения с предварительной полной очисткой результатов предыдущей сборки (очищаются папки App\target и App\lib, запускается очистка результатов сборки дочерних модулей).

Следует отметить, что указанные выше команды предварительно запускают сборку дочерних модулей приложения JepRiaShowcase.

## **Сборка модуля service-rest приложения JepRiaShowcase средствами Maven-a.**

Модуль service-rest представляет собой самостоятельное web-приложение (web-сервис), поэтому может потребоваться его отдельная сборка. Для сборки модуля service-rest средствами Maven-а предлагается следующая последовательность действий:

* 1. Обновить бинарный репозиторий. Убедиться, что бинарный репозиторий содержит следующий Maven плагин:  
       
     snapshotupdate-maven-plugin-<some version>.jar
  2. Скачать/обновить приложение JepRiaShowcase (исходники) из ветки master репозитория  
     <https://github.com/Jepria/jepria-showcase>  
     На момент написания инструкции также была доступна копия приложения JepRiaShowcase на SourceForge (SVN, ветка TRUNK):  
     <https://svn.code.sf.net/p/javaenterpriseplatform/svn/Module/JepRiaShowcase/Trunk>
  3. Открыть консоль, перейти в папку service-rest приложения JepRiaShowcase, убедиться в наличии в папке service-rest файла pom.xml и запустить сборку модуля одной из следующих команд:
* mvn package – сборка war-файла модуля (будет расположен в папке service-rest\target).
* mvn install – сборка war-файла модуля с размещением в локальном репозитории Maven-а. Важно! Название артефакта и путь его размещения будут соответствовать значениям параметров <artifactId> и <version> из файла pom.xml независимо от значения параметра <finalName> в секции <build>. При сборке модуля подобным образом он может быть использован в качестве зависимости для других проектов. Если же не предполагается использовать модуль в виде зависимости, то предпочтительнее собрать его предыдущей командой.
* mvn clean package/mvn clean install – аналоги предыдущих команд с предварительной очисткой результатов предыдущей сборки.

## **Развёртывание приложения JepRiaShowcase на Tomcat средствами Maven-а.**

Приложение JepRiaShowcase целиком (включая модуль service-rest) и модуль service-rest в отдельности разворачиваются на Tomcat средствами Maven-а абсолютно идентично. Для развёртывания предлагается следующая последовательность действий:

* 1. Для решения проблемы “захвата ресурсов” Tomcat-ом и, как следствие, некорректной работы установочного Maven-плагина необходимо включить опцию antiJARLocking или antiResourceLocking (в зависимости от версии Tomcat) для элемента <Context> в конфигурационном файле Tomcat-а context.xml.
  2. Подкорректировать pom.xml файл приложения JepRiaShowcase (или pom.xml модуля service-rest, если необходимо развернуть только service-rest), указав параметры подключения к серверу Tomcat. Необходимо отредактировать элемент с настройкой плагина tomcat7-maven-plugin. Пример настройки плагина:  
       
      <build>  
     …  
      <plugins>  
     …  
      <plugin>  
      <groupId>org.apache.tomcat.maven</groupId>  
      <artifactId>tomcat7-maven-plugin</artifactId>  
      <version>2.2</version>  
      <configuration>  
      <update>true</update>  
      <url>http://localhost:8080/manager/text</url>  
      <username>admin</username>  
      <password>123456</password>  
      <failOnError>true</failOnError>  
      <path>/JepRiaShowcase</path>  
      </configuration>  
      </plugin>  
     …  
      </plugins>  
     …  
      </build>
  3. Открыть консоль, перейти в папку App приложения JepRiaShowcase (или в папку App\service-rest, если разворачивается только модуль service-rest), убедиться в наличии в текущей папке файла pom.xml и воспользоваться одной из следующих команд:
* mvn clean tomcat7:deploy – сборка war-файла приложения (или модуля) с предварительной очисткой результатов предыдущей сборки и развёртывание его на Tomcat. Указанная команда может быть использована как для первоначального развёртывания приложения (или модуля), так и для его переустановки (в настройках плагина tomcat7-maven-plugin в pom.xml необходимо задать конфигурационный параметр <update>true</update>). Также следует отметить, что Maven плагин tomcat7-maven-plugin позволяет разворачивать приложение не только на Tomcat 7й версии, но и на более поздние версии.
* mvn tomcat7:deploy – аналог предыдущей команды без предварительной очистки результатов предыдущей сборки.
* mvn tomcat7:undeploy – удаление приложения (или модуля) с Tomcat.
* mvn tomcat7:deploy-only – развёртывание приложения на Tomcat без предварительной сборки. Приложение должно быть собрано предварительно, war-файл приложения должен размещаться в подпапке target проекта.

## **Очистка временных файлов предыдущей сборки приложения JepRiaShowcase.**

Очистка временных файлов предыдущей сборки приложения JepRiaShowcase осуществляется стандартным для Maven-а образом, однако присутствуют некоторые особенности, связанные с многомодульной архитектурой приложения. Для очистки временных файлов предыдущей сборки рекомендуется воспользоваться следующими командами (текущей папкой консоли должна быть папка App приложения JepRiaShowcase с файлом pom.xml):

* mvn clean – удаление папки App\target и очистка папки App\lib (без её удаления). То есть в данном случае удаляются только результаты предыдущей сборки приложения JepRiaShowcase без удаления результатов предыдущей сборки дочерних модулей.
* mvn clean-all – дополнительно к результату выполнения предыдущей команды запускается удаление результатов предыдущей сборки дочерних модулей приложения JepRiaShowcase (gwt и service-rest).

## **Очистка временных файлов предыдущей сборки модуля service-rest приложения JepRiaShowcase.**

Очистка временных файлов предыдущей сборки модуля service-rest осуществляется стандартным для Maven-а образом. Для очистки временных файлов предыдущей сборки рекомендуется воспользоваться следующими командами (текущей папкой консоли должна быть папка App\service-rest с файлом pom.xml модуля service-rest):

* mvn clean – удаление папки service-rest\target с результатами предыдущей сборки. Если же требуется очистить папку service-rest\target с сохранением war-файла модуля service-rest, то необходимо раскомментировать в pom.xml файле модуля в секции <build> в подсекции <plugins> настройку плагина maven-clean-plugin:  
    
   <!--plugin>  
   <artifactId>maven-clean-plugin</artifactId>  
   <version>3.1.0</version>  
   <configuration>  
   <excludeDefaultDirectories>true</excludeDefaultDirectories>  
   <filesets>  
   <fileset>  
   <directory>${project.build.directory}</directory>  
   <excludes>  
   <exclude>\*.war</exclude>  
   </excludes>  
   </fileset>  
   </filesets>  
   </configuration>  
   </plugin-->
* mvn clean package clean – сборка war-файла модуля с очисткой содержимого папки service-rest\target. Команда имеет смысл совместно с настройкой из предыдущего пункта (папка service-rest\target будет очищена с сохранением war-файла модуля).

## **Сборка модуля service-rest приложения JepRiaShowcase средствами Maven-a в offline-режиме.**

На текущий момент отсутствует поддержка сборки в offline-режиме для всего приложения JepRiaShowcase (многомодульная структура не позволяет напрямую задействовать стандартный механизм), однако модуль service-rest в отдельности может быть собран. Для применения Maven-а в offline-режиме рекомендуется воспользоваться следующими командами (текущей папкой консоли должна быть папка App\service-rest с файлом pom.xml модуля service-rest):

* mvn dependency:go-offline – подготовка проекта для offline-сборки (предварительное скачивание в локальный кэш Maven-а всех необходимых зависимостей).
* mvn -o clean package – непосредственно сборка проекта по зависимостям из локального кэша Maven-а.

## **Возможные проблемы при сборке приложения средствами Maven-а.**

В процессе перевода сборки JepRia проектов c Ant-а на Maven были выявлены следующие возможные проблемы:

1. Временная недоступность сервера, который выступает в качестве зеркала центрального репозитория Maven-а (при сборке с компьютера в корпоративной сети).
2. Наличие в бинарном репозитории библиотек, путь размещения которых не соответствует соглашению о размещении артефактов Maven-а (механизм разрешения зависимостей Maven-а для таких артефактов работать не будет). Рассмотрим проблему чуть более подробно. Maven при разрешении зависимостей (локальных из бинарного репозитария) осуществляет поиск библиотек (jar-ников) по строго сформированному пути  
     
   <путь, соответствующий group id>\<название артефакта>\<версия артефакта>\<название артефакта>-<версия артефакта>.jar  
     
   Однако не все артефакты бинарного репозитория соответствуют указанному соглашению. Пример валидного пути для поиска артефакта:  
     
   ${BIN\_HOME}/deploy/com/technology/jep/jepria/jepria/${JEPRIA\_VERSION}/jepria-${JEPRIA\_VERSION}.jar  
     
   Пример невалидного пути:  
     
   ${BIN\_HOME}/deploy/com/technology/jep/jepria/jepria/${JEPRIA\_VERSION}/jepria-dependency-${JEPRIA\_VERSION}.jar  
     
   Для корректного разрешения зависимости Maven предполагает такой путь:  
     
   ${BIN\_HOME}/deploy/com/technology/jep/jepria/jepria-dependency/${JEPRIA\_VERSION}/jepria-dependency-${JEPRIA\_VERSION}.jar  
     
   В качестве временного решения описанной проблемы можно продублировать артефакты в бинарном репозитории по корректному пути.
3. Отсутствие в центральном репозитории Maven-а требуемых зависимостей при сборке с компьютера за пределами корпоративной сети. В качестве решения необходимо указать соответствующий дополнительный репозиторий в секции <profiles> конфигурационного файла .m2\settings.xml.
4. Отсутствие в корпоративном зеркале центрального репозитория требуемых зависимостей при сборке с компьютера в корпоративной сети. В качестве временного решения указанной проблемы предлагается разместить требуемые зависимости в бинарном репозитории (по корректному пути).