**Инструкция по сборке модуля service-rest приложения JepRiaShowcase средствами Maven-a.**

Для сборки модуля service-rest средствами Maven-а предлагается следующая последовательность действий:

1. Установить и сконфигурировать Maven:
   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.

* 1. 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);
  2. 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>rusfinance</id>  
      <mirrorOf>central</mirrorOf>  
      <url>http://vmskappint2:8081/repository/maven-central/</url>  
     </mirror>
  3. подключить в качестве дополнительного репозитория для поиска зависимостей и плагинов локальную папку с нашим 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 SVN папками  
     <http://svn.code.sf.net/p/javaenterpriseplatform/svn/Bin>  
     или  
     svn://srvsvn/JEP/Bin  
     Системная переменная среды ${BIN\_HOME} ссылается на указанный бинарный репозиторий, например ${BIN\_HOME} = c:\SVN\JEP\Bin\.
  4. 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 располагается в папке с настоящим документом.

1. Обновить наш бинарный репозиторий.
2. Скачать/обновить приложение JepRiaShowcase (исходники) из ветки TRUNK репозитория  
     
   <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 clean package – сборка war-файла модуля (будет расположен в папке service-rest\target).
* mvn clean 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> файла pom.xml независимо от значения параметра <finalName> в секции <build>.

При необходимости развернуть модуль service-rest на Tomcat средствами Maven-а могут быть задействованы следующие команды:

* mvn clean tomcat7:deploy – сборка war-файла модуля и развёртывание его на Tomcat (параметры подключения к Tomcat задаются в файле pom.xml в настройках плагина tomcat7-maven-plugin). Указанная команда должна использоваться при первом развёртывании приложения: если приложение уже развёрнуто, Maven выдаст ошибку. 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:undeploy – удаление модуля с Tomcat.
* mvn clean package tomcat7:redeploy – сборка war-файла модуля и переустановка его на Tomcat.

Для очистки временных файлов предыдущей сборки рекомендуется воспользоваться следующими командами:

* mvn clean – удаление папки service-rest\target с результатами предыдущей сборки. Если же требуется очистить папку service-rest\target с сохранением war-файла приложения, то необходимо раскомментировать в pom.xml файле 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).

Другие команды могут быть найдены в Интернете.

Возможные проблемы при сборке приложения средствами Maven-а:

1. Временная недоступность сервера vmskappint2, который выступает в качестве зеркала центрального репозитория Maven-а.
2. Наличие в нашем 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 нашего 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 (т.е. в нашем корпоративном зеркале центрального репозитория) требуемых зависимостей. В качестве временного решения указанной проблемы предлагается разместить требуемые зависимости в нашем бинарном репозитории (по корректному пути).