

# Развертывание приложения с помощью Java Web Start

Blandger

В данной статье я хочу рассказать о своем опыте по развертыванию и последующему "автоматическому" обновлению версий GUI клиента с помощью использования технологии Java Web Start в среде Windows. Java Web Start может быть использован в операционных системах: Windows(95, 98, ME, NT, W2K, XP), Linux, Unix (Solaris) и не так давно, в Macintosh OS X. Для развертывания приложений используется HTTP протокол, поэтому можно воспользоваться любым HTTP сервером. Но для того, чтобы использовать ВСЕ возможности, предоставляемые технологией Java Web Start, такие как "версионность" Java-приложений и "наращивающихся обновлений" (incremental update), необходимо использовать Web-сервер, поддерживающий сервлеты (Servlets) или CGI-скрипты. Мы воспользуемся сервером приложений, в нашем случае это JBoss 3.x.x, и обычно "встроенным" в него Web-контейнером — Apache TomCat. В статье я расскажу, что необходимо сделать для настройки поддержки JNPL протокола на сервере JBoss 3.x.x, использующим в качестве Web-контейнера именно Apache TomCat, поставляемый вместе с JBoss.

## 1. Введение

Что такое Java Web Start? Это небольшая, бесплатно распространяемая программа на клиентском ПК ассоциированная с веб-браузером. Когда пользователь щелкает в браузере на HTML странице ссылку, указывающую на специальный JNLP (Java Network Launching Protocol) файл запуска Java-приложения, это приводит к запуску Java Web Start, который в свою очередь автоматически скачивает файлы приложения с Web-сервера, кэширует их и запускает описанное Java-приложение. Java Web Start идет в стандартной инсталляции как JRE 1.4.x так и в JDK 1.4.x. Стоит сказать, что Java Web Start — это кроме того и набор технологий + API, который позволяет очень

легко решить вопрос установки Java-приложений (также и Applet-ов) и их последующего "автоматического" обновления на любом ПК в корпоративной (и не только корпоративной) среде. После "единичной" настройки самого Java Web Start на клиентском ПК и установки вашего Java-приложения, все необходимые для работы JAR библиотеки кэшируются на клиентском ПК. Последующие обновления приложения (при обновлении функциональности приложения, при устранении в коде ошибок) на клиентском ПК выполняются при запуске приложения АВТОМАТИЧЕСКИ. Т.е. Java Web Start позволит вам не заботиться об обновлении Java-приложения на локальных ПК, и выполнит это "самостоятельно и автоматически", проверяя и скачивая новые версии НЕ ТОЛЬКО написанных ВАМИ модулей, но также и обновленные версии используемых клиентским приложением сторонних библиотек. При этом на клиенте происходит обновление только тех файлов библиотек, которые изменились на сервере. Когда код вашего клиентского приложения обновился и вам требуется обновить его на локальных ПК, вам потребуется всего лишь положить новые версии JAR-библиотек приложения на сервер приложений (в WAR-приложение). При запуске установленного и кэшированного на клиенте приложения, Java Web Start выполняет сравнение локальных версий файлов и файлов на сервере. При изменении файлов на сервере, выполняется выборочное скачивание ТОЛЬКО изменившихся JAR файлов.

## **2. Требования к Java-приложениям и настройки на клиентском ПК**

Так как работа Java Web Start основана на использовании JNLP-протокола, то выполнить настройки необходимо как на стороне СЕРВЕРА, так и на стороне КЛИЕНТА.

Для установки Java-приложений на локальном ПК нам понадобится установленный Java Web Start (Application Manager) и веб-браузер. Браузер требуется только для первоначального запуска Java-приложения и после запуска может быть закрыт, в то время как приложение будет продолжать работать. В качестве браузера лучше сначала использовать IE, т.к. он работает корректно сразу. Также можно воспользоваться и другими браузерами (Mozilla, Opera 7.x), но для этого необходимо выполнить в них небольшие настройки. Как настроить Opera 7.x для правильной работы с JNLP

## *Развертывание приложения с помощью Java Web Start*

файлами, я написал в самом конце этой статьи, аналогичным образом должны настраиваться и другие браузеры.

Если Java-приложение запускается часто, то в среде Windows можно с помощью Java Web Start создать стандартный "ярлык приложения" на рабочем столе и запускать Java-приложение НЕ ИСПОЛЬЗУЯ браузер, а пользуясь только ярлыком. Также можно запускать Java-приложение из командной строки.

Как уже было сказано, Java Web Start всегда доступен как при установке JRE 1.4.x, так и при установке JDK 1.4.x, поэтому нам ничего не остается сделать как воспользоваться его возможностями. К сожалению для версии JDK/JRE 1.3 его нужно устанавливать отдельно. Но я бы НЕ советовал этого делать, т.к. на мой взгляд он сыроват и лучше всего перевести код вашего приложения на версию JDK 1.4, тем более что GUI клиент прекрасно взаимодействует с сервером под JDK 1.3. Проблем с передачей сериализованных объектов между версиями JDK 1.3 <-> 1.4 замечено не было.

Кроме этого Java Web Start предъявляет определенные требования к написанному клиентскому Java-приложению. Приложение должно поставляться как набор JAR-файлов, все ресурсы приложения, такие как изображения, конфигурационные файлы, Native библиотеки (DLL, SO), необходимо включать в JAR-файлы. Ресурсы в коде должны получаться с помощью `ClassLoader.getResource` или подобных методов. Приложение запускается на клиентском ПК с правами доступа, выданных для "стандартной песочницы" (sandbox), со всеми вытекающими последствиями. Поэтому если вам необходим неограниченный доступ к локальным файлам — потребуются дополнительные настройки и подписывание библиотек кода с помощью сертификата. Также для хранения локальных клиентских настроек в JWS имеется специальное `PersistenceService` API, которое чем-то похоже на "cookies" и позволяет безопасным способом хранить локальные настройки на ПК. Кроме этого, есть еще другие API — `BasicService`, `ClipboardService`, `DownloadService`, `FileOpenService`, `FileSaveService`, `PrintService`.

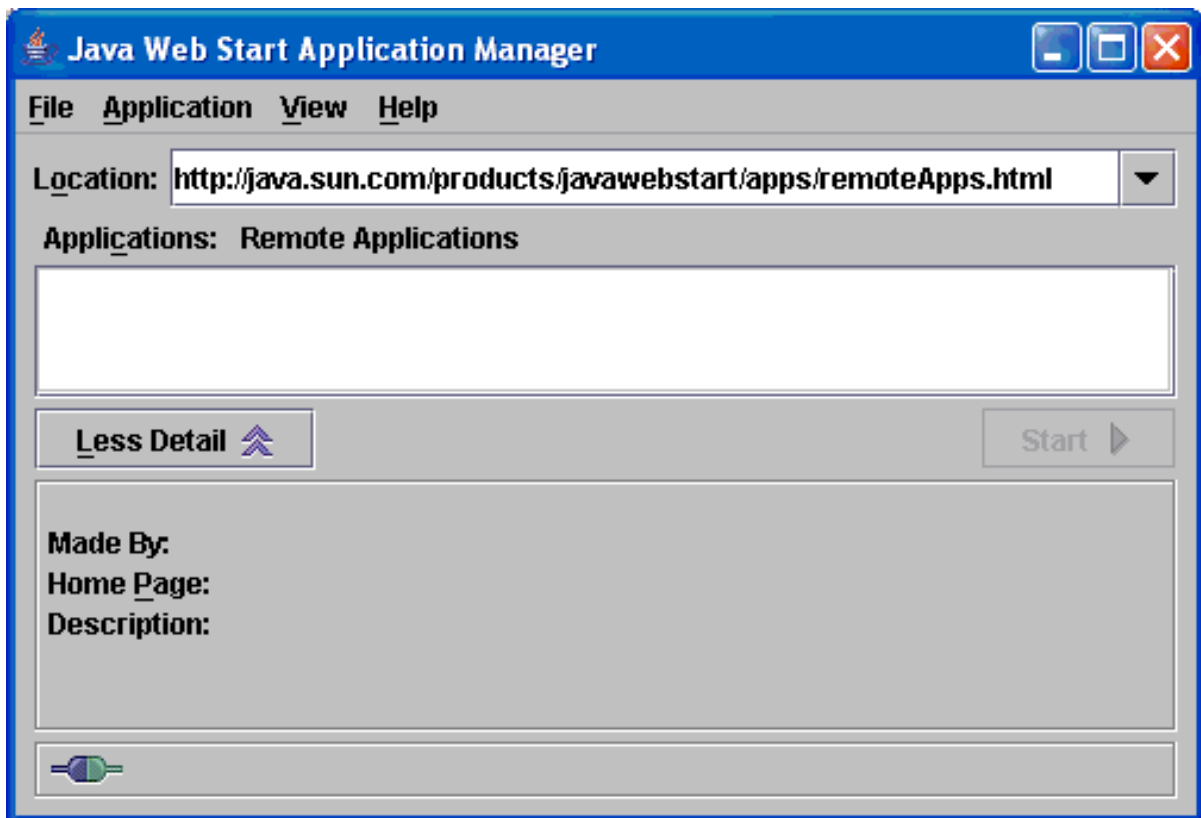
Если вы никогда не запускали и не видели на "рабочем столе" ярлыка Java Web Start, который изображается обычно такой иконкой,

## Развертывание приложения с помощью Java Web Start



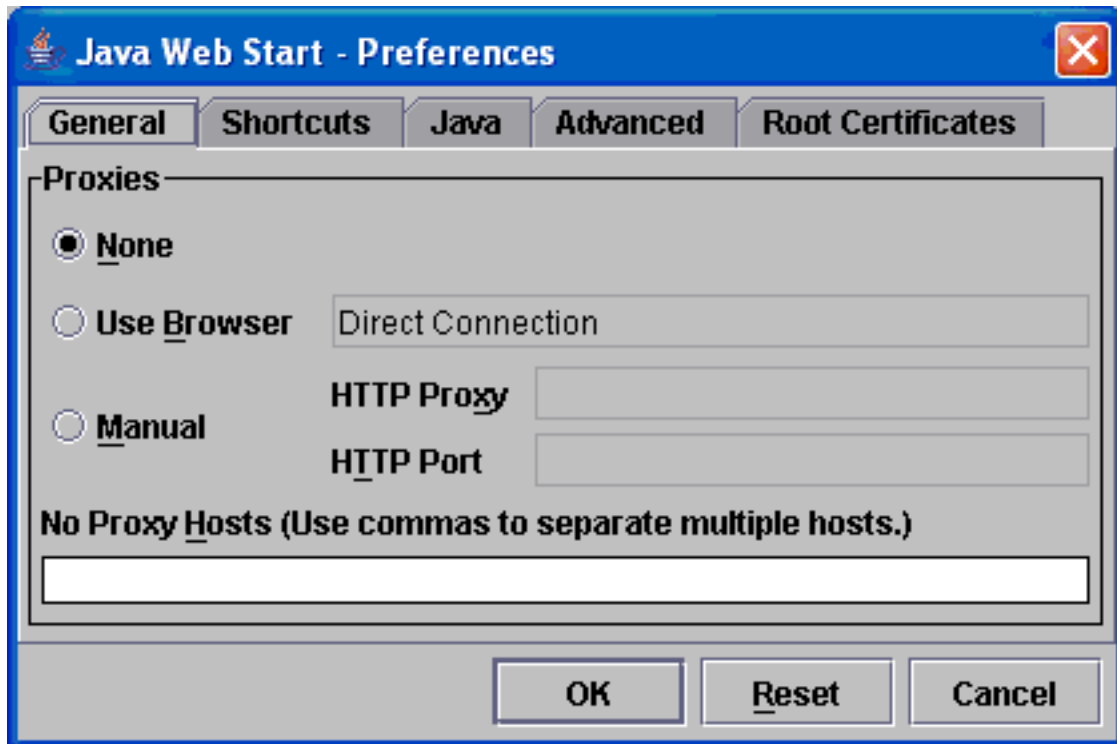
тогда вам необходимо найти данное приложение в каталоге, где установлен JDK (или JRE). Для установленной JDK 1.4 это будет например файл: `... \j2sdk1.4.2 \jre \javaws \javaws.exe`, который находится в каталоге Java Web Start (`.. \javaws`). В этом же каталоге находятся и другие DLL файлы, необходимые для его корректной работы. В случае установки ТОЛЬКО JRE данный запускаемый файл можно найти в каталоге — `... \Program Files \Java \j2re1.4.xx \javaws \javaws.exe`

Если вы не видите этого ярлыка, то необходимо найти указанный исполняемый файл и запустить его. После запуска вы должны увидеть окно Java Web Start Application Manager:



## Развертывание приложения с помощью Java Web Start

Если вы запустили на локальном ПК клиента Java Web Start, то можно считать, что этого достаточно для того, чтобы устанавливать клиентские Java-приложения. Но я все-таки хочу обратить ваше внимание на дополнительные параметры настройки JWS. Откройте в приложении JWS Application Manager: File -> Preferences, закладка General.



На ней можно указать прокси-сервер. Эта настройка зависит от настроек вашей сети, которые необходимо уточнить у системного администратора. В моем случае их нужно было "выключить", поставив значение — None, а иначе загрузка клиентского приложения либо не происходила, либо выполнялась очень долго через прокси-сервер. Надо сказать, что в локальной 100 Мбитной сети первоначальная загрузка и кэширование библиотек выполняется довольно быстро (от нескольких секунд до нескольких минут) и зависит от объема всех библиотек, входящих в ваше приложение.

Прежде чем мы продолжим дальнейшую настройку Java Web Start, я хочу сказать, что для более подробного изучения возможностей JWS, необходимо обратиться к документации разработчика на сайте Sun —

<http://java.sun.com/products/javawebstart/developers.html>.

Подробности можно найти также в PDF спецификации — JAVA™ NETWORK LAUNCHING PROTOCOL & API SPECIFICATION (JSR-56) VERSION 1.0.1 более новая версия протокола уже 1.2, по адресу — <http://java.sun.com/products/javawebstart/download-spec.html>

Кроме этого, если вы захотите, используя эту статью, самостоятельно настроить деплоймент приложений, то вам необходимо взять на сайте Sun архив, предназначенный для разработчика. Этот архив доступен для загрузки как "Java Network Launching Protocol (JNLP) Developer's Pack" по адресу — <http://java.sun.com/products/javawebstart/download-jnlp.html> Сейчас это файл javaws-1\_2-dev.zip размером около 160 Кб. Он содержит минимально необходимую документацию о настройке JNLP на клиенте и сервере, а также JAR-архивы классов сервлетов (jardiff.jar, jnlp-servlet.jar, jnlp.jar), которые будут необходимы на сервере.

На странице <http://java.sun.com/products/javawebstart/reference/index.html> можно найти ссылки на другие разделы, связанные с использованием JWS.

API Specifications

Documentation

Code Samples and Apps

Technical Articles and Tips

Partner Program

### **3. Создание JNLP файла, описывающего ваше клиентское приложение, его библиотеки и другие параметры**

Для того, чтобы Java Web Start мог знать какие именно файлы необходимы для запуска и работы вашего клиентского приложения, необходимо создать специальный JNLP файл, имеющий XML формат. Данный файл будет помещен на сервере JBoss в Web-приложение, предназначенное для выполнения деплоя GUI.

Я приведу здесь краткий и простейший пример JNLP файла, использованного для

## Развертывание приложения с помощью Java Web Start

деплоймента, и дам небольшие пояснения о тегах файла и их значениях. Для более детального описания всех параметров и всех возможностей данной технологии, я очень рекомендую вам обратиться к документации разработчика на сайте Sun Microsystems.

```
<?xml version="1.0" encoding="Windows-1251"?>
    .....
<jnlp
    spec="1.0+"      <!-- Номер JNLP спецификации -->

    codebase="http://localhost:8080/application"
    <!--URL по которому находится JNLP файл, можно написать в виде
        codebase="$$codebase" -->

    href="application.jnlp" >
    <!--название JNLP файла-дескриптора нашего приложения
        можно написать в виде href="$$name" -->

    <information>
        <title>Corporate GUI client</title>
        <vendor>Company ZZZ</vendor>
        <description>Company's corporate client</description>
    </information>
    <resources>
        <j2se version="1.3+"/> <!--Указание необходимой версии JDK приложения-->
        <jar href="main_gui.jar" main="true"/><!--Файлы нашего клиентского приложения-->
        <jar href="main_gui_lib.jar" />

        <jar href="jboss-client.jar"/>
        <jar href="jboss-common-client.jar"/> <!--Перечисление всех файлов сторонних-->
        <jar href="jboss-j2ee.jar"/>          <!--библиотек, необходимых для запуска-->
        <jar href="jbossmq.jar"/>            <!--нашего приложения -->
        <jar href="jbossx-client.jar"/>
        <jar href="jnp-client.jar"/>
        <jar href="xercesImpl.jar"/>
        <jar href="xmlParserAPIs.jar"/>
        <property name="java.naming.provider.url" value="localhost:1099"/>
            <!-- Указание свойства, которое используется нашим приложением -->
    </resources>
    <application-desc main-class="com.my_company_name.client.Application" />
```

## Развертывание приложения с помощью Java Web Start

```
<!-- Название класса с main() точкой запуска -->
</jnlp>
```

`codebase="http://localhost:8080/application"` — указывает на параметр "базы кода" по которому мы будем хранить все необходимые библиотеки, как JAR файлы нашего приложения, так и JAR файлы "сторонних библиотек". Данный параметр можно заменить "специальной переменной", фактическое значение которой jnlp-сервлет поставит самостоятельно при обработке запроса.

`href="application.jnlp"` — название JNLP файла-дескриптора, который описывает наше приложение. Данный параметр также можно заменить "специальной переменной", фактическое значение которой jnlp-сервлет поставит при обработке запроса.

В разделе ресурсов, есть указание использования JRE версии 1.3 и более новых — `<j2se version="1.3+"/>`. А вообще, элемент `<resources>` может содержать 6 различных подэлементов, таких как: `jar`, `nativelib`, `j2se`, `property`, `package` и `extension`. Подробности и правила можно найти в документации.

`<jar href="main_gui.jar" main="true"/>` — Указание библиотеки, в которой находятся классы нашего приложения, при этом параметр `main="true"`, указывает что данный JAR архив содержит запускаемый класс GUI приложения.

`<jar href="jboss-client.jar"/>` — далее перечислены все необходимые библиотеки, которые будут получены с сервера и кэшированы на клиенте

`<property name="java.naming.provider.url" value="localhost:1099"/>` — так мы можем перечислить все, передаваемые в качестве параметров запуска приложению свойства, которые получаются вызовом `System.getProperty(...)`

`<application-desc main-class="com.my_company_name.client.Application" />` — указание полного запускаемого класса. JWS также поддерживает запуск Applet-ов. В этом случае вместо тэга `<application-desc>` используется тэг `<applet-desc>`. Принцип написания и параметры — смотрите в документации.

Что касается элемента `<information>` JNLP файла. В данном тэге значения

## Развертывание приложения с помощью Java Web Start

подэлементов <title> и другие, наверное, пока что лучше указывать на английском языке. В последней версии Java Web Start (1.2) из версии JDK 1.4.2\_04-b05 название на русском языке в JNLP файле, вызвали ошибку при конвертировании русских букв. Ошибка наблюдалась в логе JBoss (server.log):

```
ERROR [org.jboss.web.localhost.Engine] Internal error:
sun.io.MalformedInputException
  at sun.io.ByteToCharUTF8.convert(ByteToCharUTF8.java:90)
  at java.io.InputStreamReader.convertInto(InputStreamReader.java:137)
  at java.io.InputStreamReader.fill(InputStreamReader.java:186)
  .....
  at com.sun.javaws.servlet.JnlpDownloadServlet.doGet(JnlpDownloadServlet.java:79)
  at javax.servlet.http.HttpServlet.service(HttpServlet.java:740)
  .....
```

JNLP файл имеет также дополнительные параметры и позволяет указывать разные ресурсы приложения в зависимости от: версии самого приложения, версии операционной системы, платформы, "локали" — т.е. поддерживает "версионность" приложений по разным критериям. В качестве ресурсов можно также указывать "native" библиотеки (например DLL, SO), используемые вашим приложением. Если вашему приложению требуется доступ к локальным файлам или другие права на локальном ПК, то для этого существует раздел <security>, который необходимо также описать. При необходимости доступа к локальным ресурсам на ПК, например файлов, все библиотеки вашего приложения должны быть подписаны сертификатом, который можно сгенерировать самостоятельно. Все подробности и правила описания можно найти в документации.

Для создания JNLP деплоймент файлов можно использовать свободно распространяемый "DeployDirector" от Sitraka Software, подробности читайте на [сайте производителя](#).

## 4. Настройка поддержки JNLP (Java Network Launching Protocol) на сервере JBoss 3.x.x с установленным Web-контейнером Apache TomCat

К сожалению, мне пока не приходилось настраивать JNLP на сервере JBoss, который использует в качестве Web-контейнера Mortbay Jetty, а только Apache TomCat. Но

посмотрев внутренности Jetty, могу сказать, что это тоже вполне возможно сделать аналогичным способом. Нужно только внимательно посмотреть и найти файлы настроек в JAR архивах данного Web-контейнера, в которые понадобится внести аналогичные добавления.

Итак, какие изменения вносят на сервере...

## **4.1. Добавление поддержки новых MIME типов в Apache TomCat**

Мы выполним данную настройку "глобально" для всего TomCat, чтобы этот MIME тип был описан для всех Web-приложений контейнера, но такого же эффекта можно добиться, если добавить дополнительные MIME типы ТОЛЬКО в web.xml отдельного web-приложения, которое будет вами сделано на сервере для деплоя клиентского ПО. Для добавления новых MIME типов, мы находим в каталоге JBoss, подкаталог в котором установлен TomCat. В случае JBoss 3.x — это скорее всего каталог: `...\jboss\catalina\`

Находим конфигурационный файл — `...\jboss\catalina\conf\web.xml`.

В последних версиях JBoss 3.2.x данный файл необходимо искать в каталоге:

`...\jboss-3.2.3\server\default\  
deploy\jbossweb-tomcat41.sar\web.xml`

В данном файле находим раздел, где описаны MIME типы, проверяем есть ли они уже в списке описанных. `jnlp`, `jar` — обычно уже есть, а вот `jardiff` — скорее всего необходимо добавить. При их полном отсутствии добавляем, например в начало списка, еще три дополнительных типа:

```
web.xml
.....
<!-- ===== Default MIME Type Mappings ===== -->
<mime-mapping>
  <extension>jnlp</extension>
  <mime-type>application/x-java-jnlp-file</mime-type>
</mime-mapping>
<mime-mapping>
  <extension>jar</extension>
```

## Развертывание приложения с помощью Java Web Start

```
<mime-type>application/x-java-archive</mime-type>
</mime-mapping>
<mime-mapping>
  <extension>jardiff</extension>
  <mime-type>application/x-java-archive-diff</mime-type>
</mime-mapping>
.....
```

Если дать "грубое пояснение", то этими действиями мы указали Web-контейнеру выполнять "специальную интерпретацию" файлов с расширениями — jnlp, jar, jardiff. После HTTP запроса (request) клиентом у контейнера файла-ресурса с одним из указанных расширений, контейнер должен поставить в заголовке HTTP ответа (response) соответствующий "Content-Type", равный — application/x-java-jnlp-file, application/x-java-archive, application/x-java-archive-diff. При этом указанное "Content-Type" значение в HTTP ответе, позволит Java Web Start на стороне клиента, обработать данный поток данных "специальным образом".

Замечание: Будьте внимательны с тем, что можно легко перепутать расширение файлов \*.jnlp, ошибочно назвав \*.jnpl, как в настройках Web-контейнера, так и в названиях файлов.

### 4.2. Создание архива Web-приложения (WAR), предназначенного для деплоя Java-приложения на локальные ПК

В каталоге JBoss, предназначенного для деплоя J2EE приложений, создадим новое Web-приложение. Его можно создать любым удобным способом, я предлагаю сделать так: пусть это будет "default" конфигурация. Внутри деплоймент каталога, создаем каталог web-приложения с названием, указанным в JNLP файле, это будет — ..\application.war

```
...\jboss-3.2.1\server\default\deploy\application.war\
```

В моем случае содержимое WAR приложения выглядит так:

```
\application.war\
  index.html      <-- Файл содержащий URL-ссылку на JNLP файл
```

## Развертывание приложения с помощью Java Web Start

```
клиентского Java-приложения
application.jnlp  <- специально написанный XML файл с расширением jnlp,
                  описывающий состав и параметры клиентского приложения

main_gui.jar     <- главные архивы классов клиентского приложения
main_gui_lib.jar

jboss-client.jar      <- библиотеки необходимые для выполнения
jboss-common-client.jar  приложения на ПК пользователя
jboss-j2ee.jar
jbossmq.jar
jbossx-client.jar
jnp-client.jar
xercesImpl.jar
xmlParserAPIs.jar

\application.war\WEB-INF\
  jboss-web.xml
  web.xml      <- файл дескриптор Web-приложения

\application.war\WEB-INF\lib\
  jardiff.jar    <- библиотеки, содержащие классы,
  jnlp-servlet.jar  необходимые для поддержки JNLP сервером, которые
  jnlp.jar        доступны из "JNLP пакета разработчика"
```

### 4.3. Добавление необходимых параметров в web.xml файл WAR-приложения

Кроме этого необходимо, описать необходимые настроечные параметры у нашего Web-приложения в файле `...\jboss-3.2.1\server\default\deploy\application.war\WEB-INF\web.xml`. Согласно документации в него нужно дописать следующие настройки, которые выделены красным цветом:

```
<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC
  "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
  "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <servlet>
    <servlet-name>JnlpDownloadServlet</servlet-name>
```

## Развертывание приложения с помощью Java Web Start

```
<servlet-class>com.sun.javaws.servlet.JnlpDownloadServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>JnlpDownloadServlet</servlet-name>
  <url-pattern>*.jnlp</url-pattern>
</servlet-mapping>

<welcome-file-list>
  <welcome-file>index.html</welcome-file>
</welcome-file-list>
<web-app>
```

Еще одно из требований для корректной работы JNLP сервлета, описанного в примере настройки — это наличие XML парсера. Для этого необходимо, что либо сам Web-контейнер был запущен с помощью JRE 1.4 , в которой парсер интегрирован, или чтобы парсер был доступен серверу как библиотека. В нашем случае, т.к. JBoss имеет в поставке XML парсер (Xerces), никаких дополнительных действий делать НЕ НАДО. В случае если ваша ситуация отличается, то добавьте парсер в Web-приложение — каталог где хранятся библиотеки приложения  
... \application.war\WEB-INF\lib\.

Теперь опишем КАК выглядит индексная страница, с которой осуществляется установка и запуск наших клиентских приложений на локальных ПК пользователей. Простейший вид страницы index.html:

```
index.html

<html>
<head><title>Клиентские приложения</title>

<meta http-equiv="content-type" content="text/html; charset=Windows-1251">
</head>

<body><h3><center>
Внутренние корпоративные клиентские приложения.
</center></h3>
<ul>
<li>Клиентское приложение 1.0.x :
<a href="application.jnlp">Клиент 1.0</a>
</li>
</ul>
```

```
</body>  
</html>
```

Страница имеет ссылку, указывающую на JNLP файл нашего приложения. По нажатию ссылки в браузере, будет происходить загрузка и запуск Java-приложения на клиентском ПК.

## 5. Пробуем запустить

Все готово к первому запуску Java-приложения. Запускаем JBoss, сначала WAR-приложение должно успешно задеплоиться. В логах вы должны увидеть приблизительно следующее:

```
server.log  
  
INFO [org.jboss.deployment.MainDeployer] Starting deployment of package:  
      file:/...../jboss/server/default/deploy/application.war/  
INFO [org.jboss.web.catalina.EmbeddedCatalinaServiceSX] deploy, ctxPath=/application,  
      warUrl=file:/...../jboss/server/default/deploy/application.war/  
      .....  
      .....  
INFO [org.jboss.deployment.MainDeployer] Successfully completed deployment of package:  
      file:/...../jboss/server/default/deploy/application.war/
```

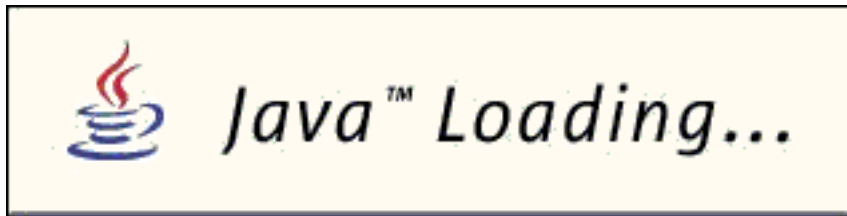
Если этого не произошло — проверьте все настройки и параметры Web-приложения.

Заходим с помощью IE на страницу нашего Web-приложения по адресу, например, <http://localhost:8080/application/>.

На странице мы должны увидеть нашу ссылку на JNLP файл в виде — <http://localhost:8080/application/application.jnlp>.

Щелкнув на ней мы должны увидеть Splash-скрин запуска Java Web Start. Для версии JDK 1.4.2 он выглядит примерно так:

## Развертывание приложения с помощью Java Web Start



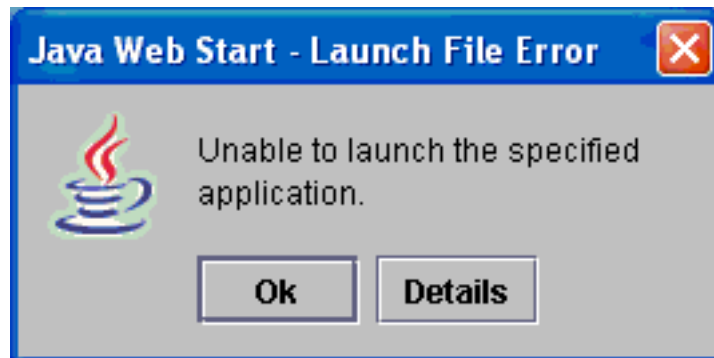
После этого на сервере в логе должны появиться записи об обращении к JNLP-сервлету примерно такого вида:

```
server.log

INFO [org.jboss.web.localhost.Engine] JnlpDownloadServlet: init
INFO [org.jboss.web.localhost.Engine] Initializing
INFO [org.jboss.web.localhost.Engine] Request: /application/application.jnlp
INFO [org.jboss.web.localhost.Engine] User-Agent: Mozilla/4.0 (compatible;
MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.4322)
INFO [org.jboss.web.localhost.Engine] DownloadRequest[path=/application.jnlp
isPlatformRequest=false]
INFO [org.jboss.web.localhost.Engine] Basic Protocol lookup
INFO [org.jboss.web.localhost.Engine] JnlpResource: JnlpResource[WAR Path:
/application.jnlp
lastModified=..... EET 2004]]
INFO [org.jboss.web.localhost.Engine] Resource returned: /application.jnlp
INFO [org.jboss.web.localhost.Engine] lastModified: ..... EET 2004
```

Если все происходит нормально, то после этого вы увидите как на ваш ПК в локальный кэш загружаются JAR библиотеки приложения, после чего приложение будет запущено. Но скорее всего в первый раз вы можете получить окно с сообщением о какой-либо ошибке. Я внес в свой JNLP файл небольшую ошибку, нарушив структуру application.jnlp XML файла. Сообщение об ошибке обычно выглядит, например, так:

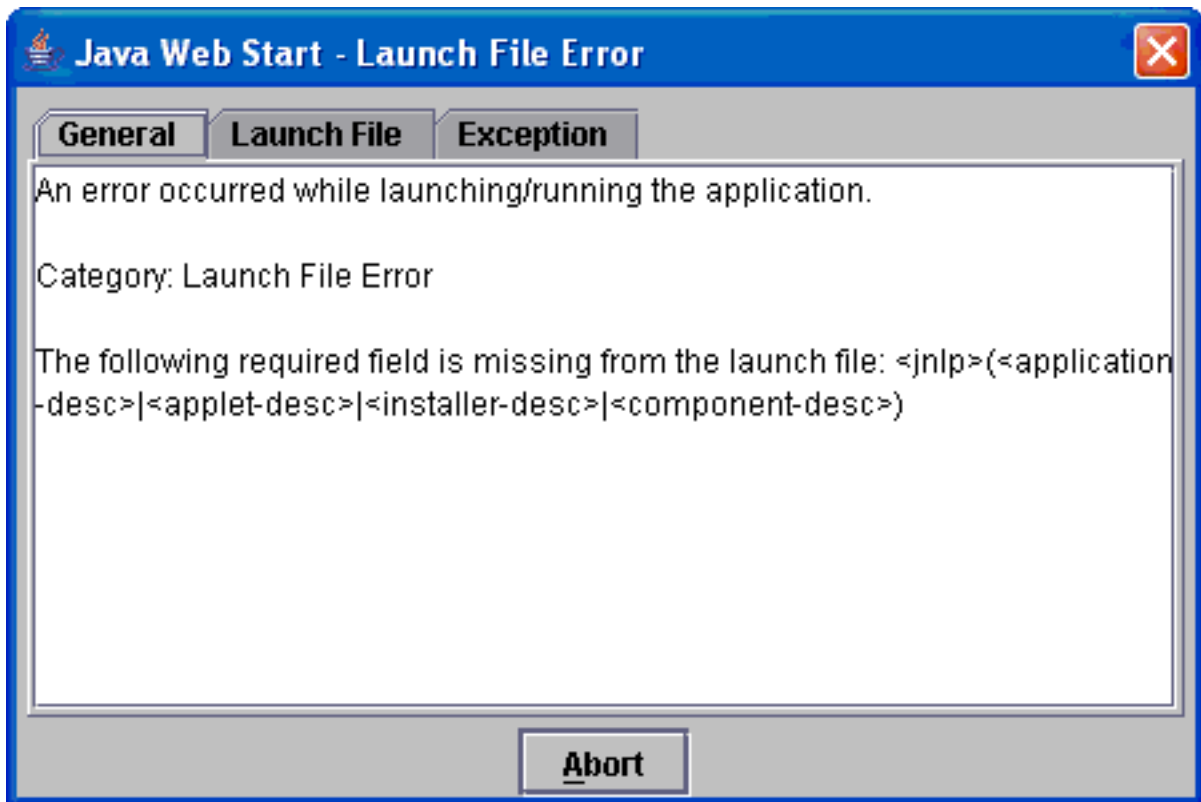
## Развертывание приложения с помощью Java Web Start



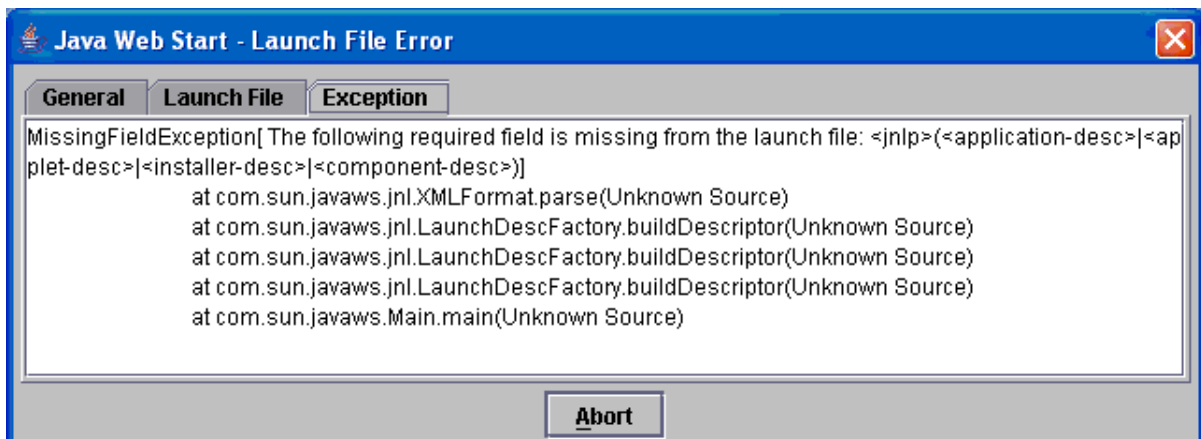
Также если ваше приложение загружается в таком "бесконечном цикле", типа "вроде бы грузиться и вроде бы совсем НЕ грузиться", а подождав некоторое время, вы получаете сообщение об "невозможности загрузить приложение" — проверьте настройки прокси-сервера, их скорее всего необходимо изменить.

Теперь мы можем посмотреть на причину данной ошибки, нажав кнопку — Details. В моем случае оно выглядело так:

## Развертывание приложения с помощью Java Web Start



Другие закладки дают дополнительную информацию о причине и характере ошибки:

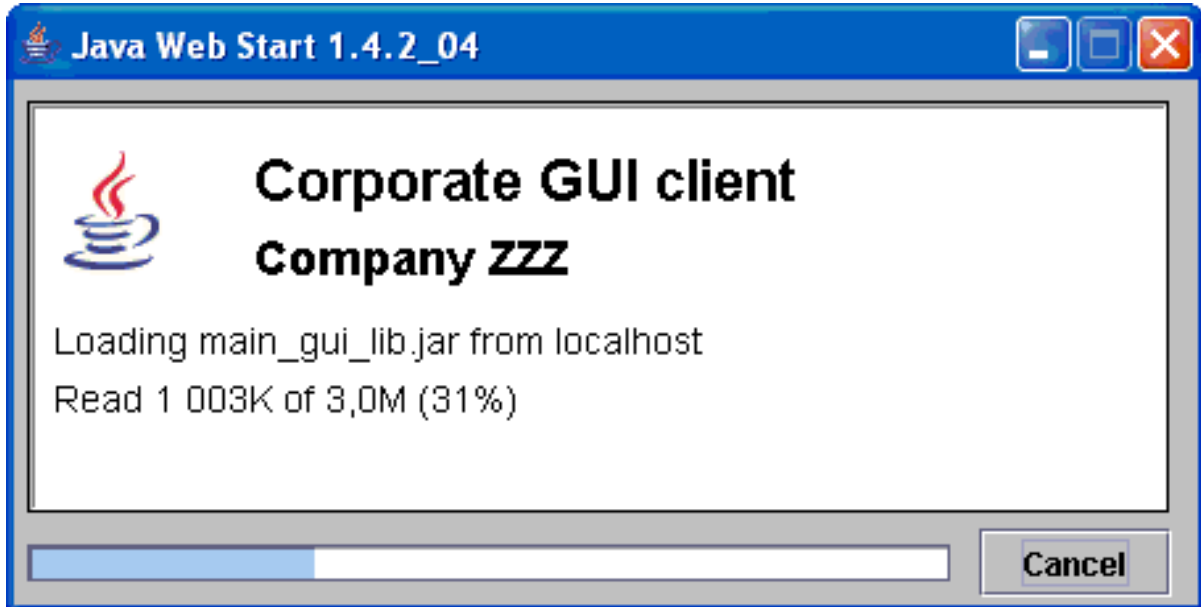


Ошибка указывает на то, что наш JNLP файл имеет не совсем корректную структуру. В вашем конкретном случае, ошибка может оказаться совсем не такой, потому что

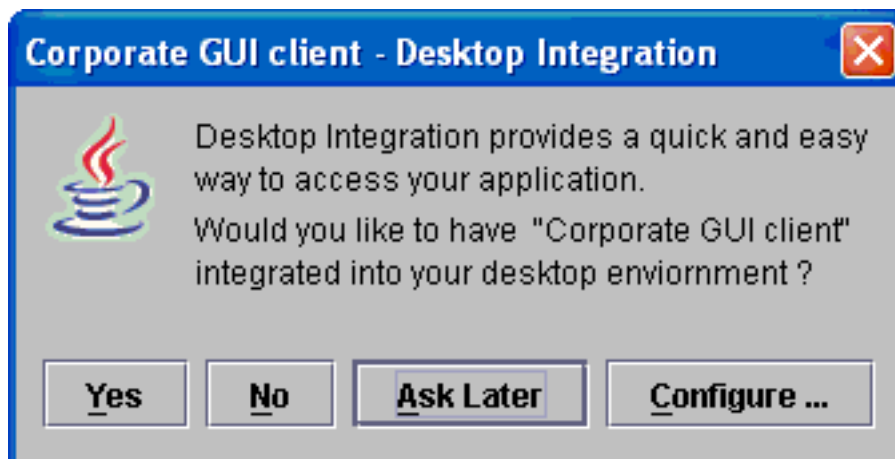
## Развертывание приложения с помощью Java Web Start

сделать ошибки всегда "хватает возможностей"...

Исправив ошибку в `application.jnlp`, который находится на сервере, мы сразу выполняем повторный запуск Java-приложения. Если все получилось, что вы увидите окно показывающее загрузку библиотек:



После чего будет предложено выполнить интеграцию вашего Java-приложения с Windows, создав "ярлык запуска" приложения на рабочем столе.



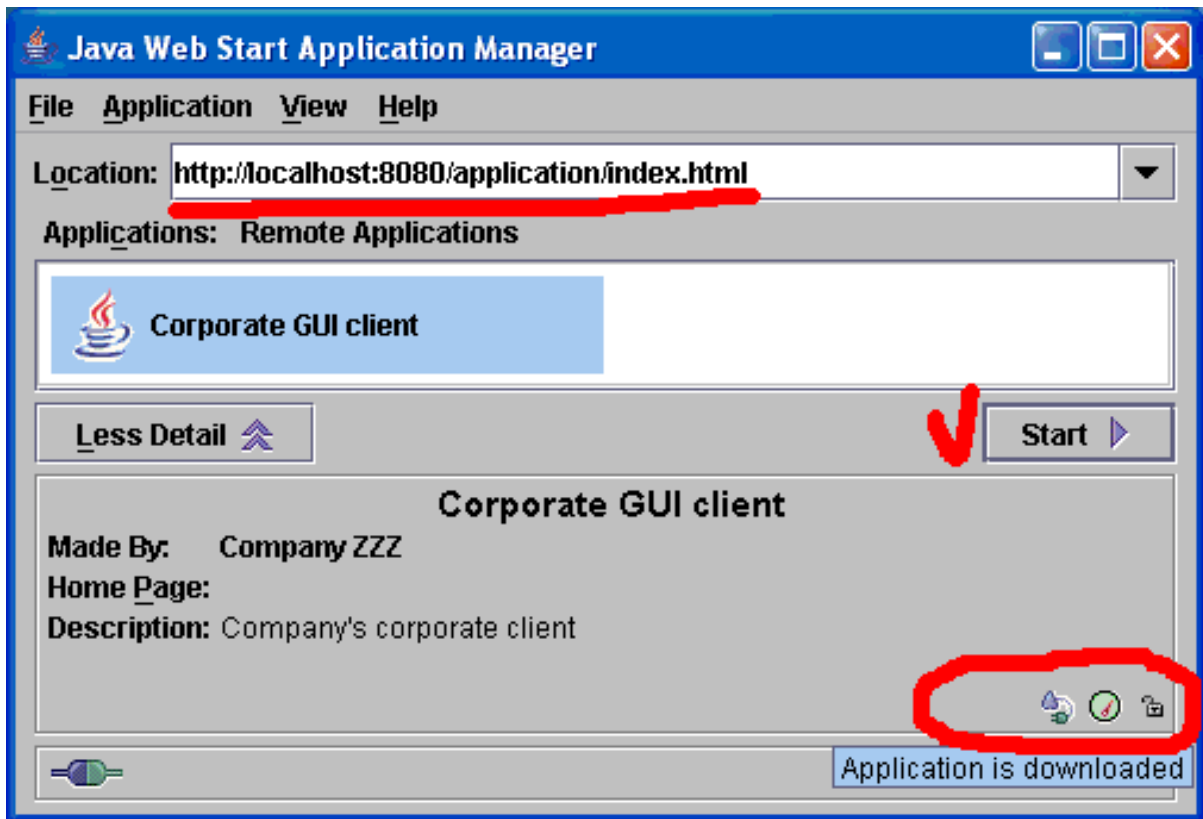
## *Развертывание приложения с помощью Java Web Start*

Если вы ответите на вопрос положительно на рабочем столе появится ярлык для запуска клиентского Java-приложения, которым можно пользоваться "напрямую", не запуская браузер.

После этого должен произойти запуск вашего GUI приложения. Если этого не происходит, то это означает, что клиентское приложение выполнилось с какой-то ошибкой. Способ обнаружения, отображения и записи ошибок в клиентском Java-приложении полностью зависит от вашей реализации. Если вы захотите логировать ошибки и/или сообщения в локальный файл, то будет необходимо выполнить дополнительные действия по настройке и подписыванию библиотек сертификатом, о которых я расскажу в другой статье. Вы также можете самостоятельно прочитать, как и что для этого нужно сделать, в документации Java Web Start.

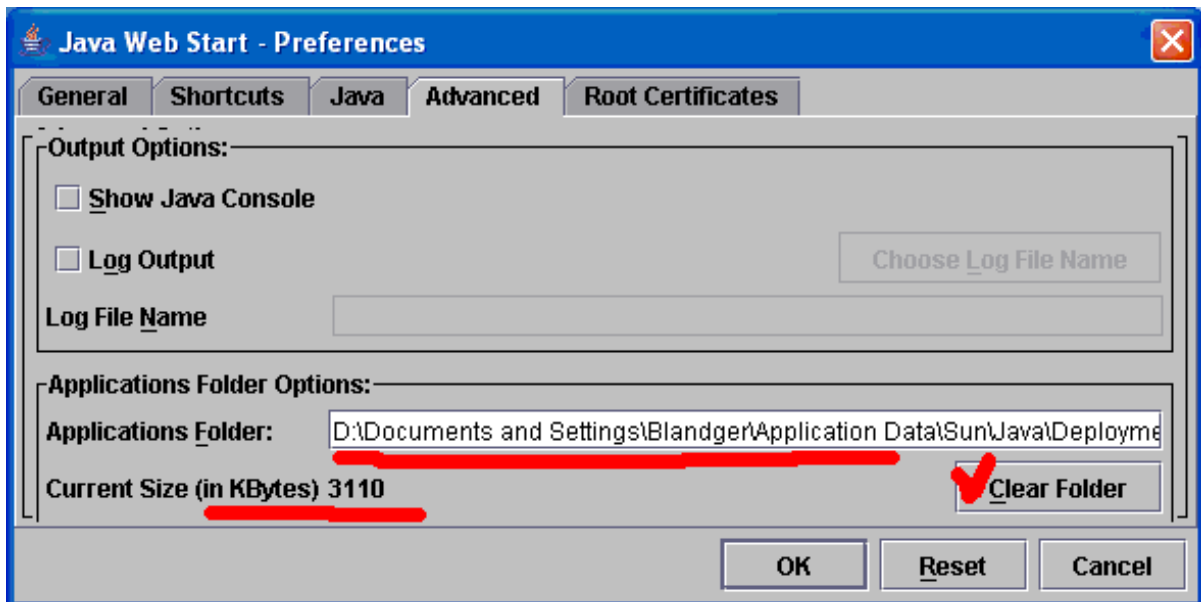
Также после успешной установки Java-приложения на клиентский ПК в JWS Application Manager появится ссылка на ваше приложение, с указанием источника. Там же с помощью иконок в правом нижнем углу JWS сообщает об доступности новых версий библиотек данного приложения.

Развертывание приложения с помощью Java Web Start



Что еще можно увидеть в настройках JWS. Например кэш, в котором хранятся Java-приложения, установленные с помощью JWS расположен в каталоге настроек пользователя:

## Развертывание приложения с помощью Java Web Start



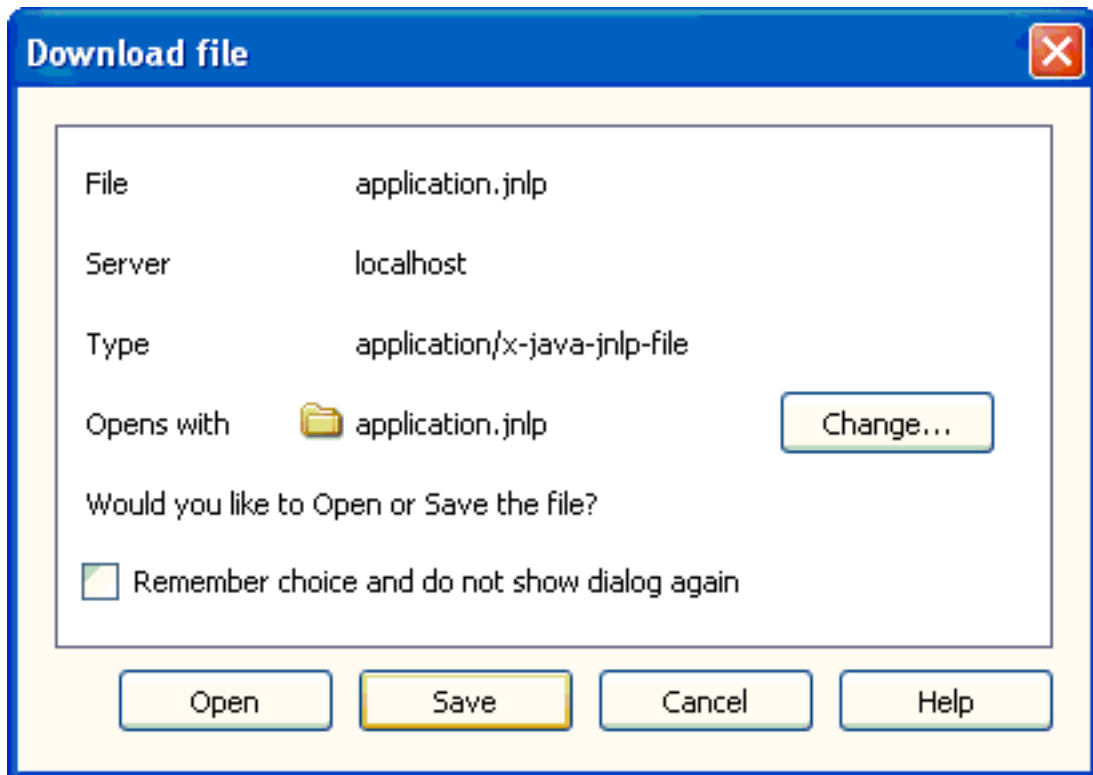
На закладке "Advanced" можно увидеть каталог кэшированного приложения, в моем случае это был `D:\Documents and Settings\Blandger\Application Data\Java\.....`. При этом также указывается текущий размер занимаемый приложением — 3110 Кб, в любое время вы можете легко очистить кэш, используя кнопку — "Clear Folder".

Теперь, если вы дочитали до этого места и все-таки не совсем поняли что происходит. Как вы думаете, что теперь необходимо сделать для обновления Java-приложения на локальных ПК пользователей после того, как мы собрали новую версию GUI приложения, или например решили использовать более свежую версию какой-то сторонней библиотеки ?

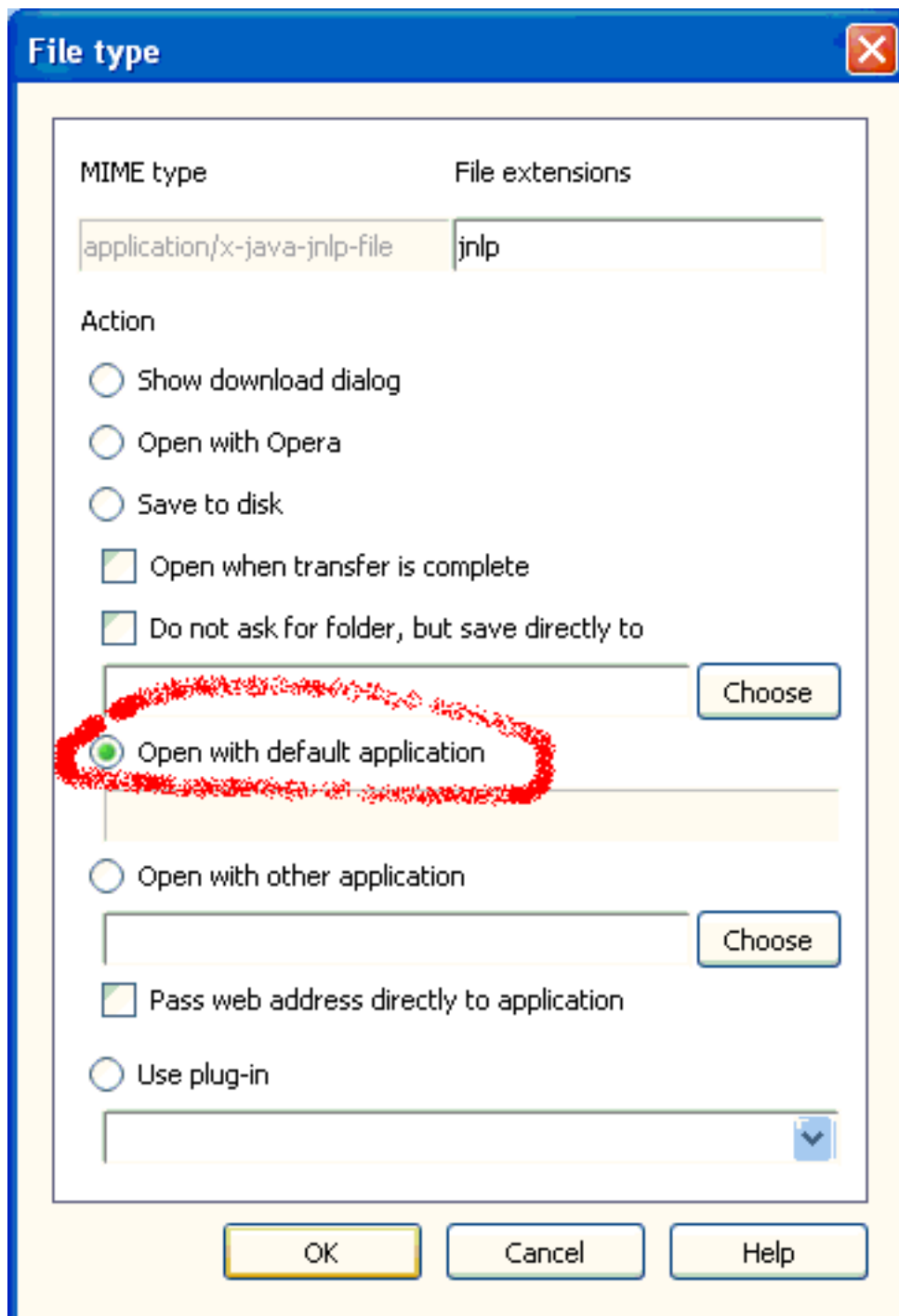
Мы должны всего лишь скопировать новые версии библиотек в каталог сервера `...\jboss-3.2.1\server\ default\deploy\application.war\`, у меня это файлы: `main_gui.jar`, `main_gui_lib.jar` (или например библиотеки — `xercesImpl.jar`, `xmlParserAPIs.jar`). При последующем запуске GUI приложения на клиентском ПК с помощью Java Web Start данные JAR файлы будут скачены на ПК, после чего приложение будет запущено с новыми версиями библиотек. Вот и все действия для обновления!

## 6. Приложение: Как настроить браузер Opera 7.x для корректной работы с JNLP файлами

Если вы попытаете запустить Java-приложение в браузере Opera кликнув на ссылке `http://localhost:8080/application/application.jnlp`, то скорее всего увидите следующее окно, в котором Opera предлагает сохранить или открыть файл.



Можно открыть файл, нажав кнопку "Open". А можно нажать кнопку "Change..." и установить обработку данного MIME-типа и расширения файлов "по умолчанию".



## *Развертывание приложения с помощью Java Web Start*

Отметьте пункт "Open with default application" и Опера будет всегда открывать JNLP файлы с помощью Java Web Start.

Автор — Yuriy Larin aka Blandger. [Оригинал статьи](#).