Toolbox: The Mini-IDE

This part of the toolbox series introduces the Mini-IDE project. It is designed to instantly setup a weblogic server together with a small sample application. It provides everything to setup and run the example in less than five minutes. Despite its simplicity, it contains everything to build and deploy a Java EE web application, relying on not more than a WLS installation. It can easily be modified and used in analysis situations or as a starting point for proof of concepts. In environments where access is restricted to shell usage, we provide a distribution method based on copy and paste via the clipboard.

1 Contents

2 Introduction

The idea of the mini-ide is to quickly set up a weblogic server domain, build and deploy a sample application and run it. We want to use only a minimum of external dependencies. Thus the mini-ide can be used on pre-production and test installations with no additional software. The mini-ide basically only requires a standard WLS installation and some common UNIX tools. We provide a script to create the WLS domain and to control it, i.e. start and stop the server and provide status information of the server process. We also provide a sample application together with a script to build and deploy the application and to run the client. Sometimes we have to work in restricted environments where shell access like Putty is the only option. No file transfer mechanisms are in place. For this situation we provide a base64 encoded archive file which is enclosed in echo commands and can be transferred to the target system using copy and paste via the clipboard.
The mini-ide is designed to be used on UNIX platforms. It is tested on MAC OSX Lion, Linux (Red Hat Enterprise Linux Server release 5.5) and also runs on Windows 7 Professional with the win-bash tools with some limitations.

3 Overview

The Mini-IDE consists of two parts, one to create and managed the WLS domain, and the other to build, deploy and run the Application. The first part only consists of the shell script control.sh which provides basic server management operations. The second part is stored in a subdirectory named Converter, the name of the application project. It contains the source code

for the sample application, the deployment descriptors and a script that provides basic IDE operations. The following figure depicts the project contents.

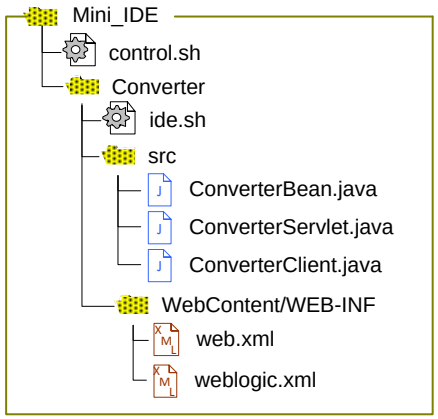

Figure 1. Directory Structure and Contents of the Mini-IDE.

The Converter project uses the WLS domain as it's test environment. It is designed in a way that it could easily be duplicated and modified so that it would built a second application to run in the same WLS domain. The following picture illustrates the runtime aspects of the Mini-IDE

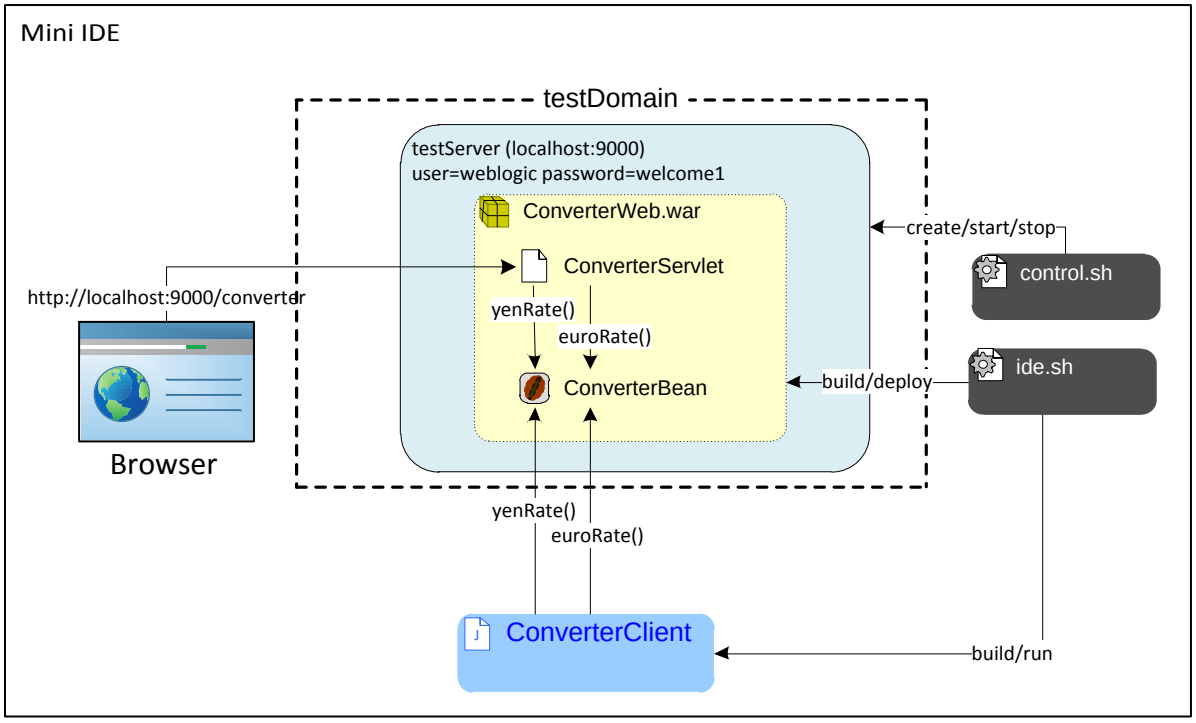

Figure 2. Overview of the runtime aspects of the Mini_IDE

The WLS domain "testDomain" is comprised of a single WLS server named "testServer" which starts a listener on port 9000. The testDomain is created with a single create command in the script control.sh. This script also sets the environment, e.g. server name, ports, classpath etc. We use the script ide.sh to build and deploy the web application ConverterWeb. This application consists of a simple stateless EJB that offers money conversion functions at its public interface. It also contains the ConverterServlet which offers a Browser interface to the ConverterBean's service. A web browser running on the local machine accesses the servlet via HTTP, for remote usage the testServer needs to be reconfigured to use the machine's IP address or DNS name as listen address. We also provide a remote Java client

"ConverterClient" which access the ConverterBean directly and prints out the result to the console. The script ide.sh contains methods to build and run the client.

4 Distribution

The distribution is contained in the file mini-ide.jar. The jar format is only used as a portable archive that can be easily unpacked on different platforms, using the Java jar tool. The mini-ide.jar is base64 encoded and enclosed in echo commands. It is contained in the file transfer-archive.txt. This text file also contains some UNIX commands to decode and unpack the base64 encoded file.
While most company networks are sealed off against different sorts of file import, it is normally possible to transfer a simple text file to the customer, e.g. via eMail. It can then be provided to the client machine that has shell access to the machines running the WLS server. We need the file transfer-archive.txt on the client machine, e.g a Windows PC. We open it with a text editor, select and copy all and paste everything to the remote shell. This process is depicted in the following picture.

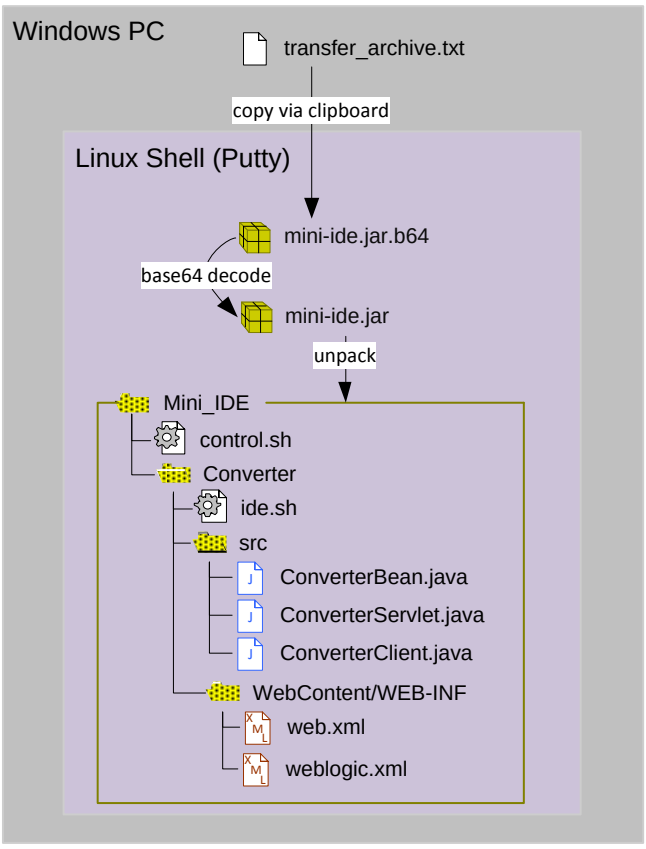

Figure 3. Distribution of the Mini-IDE to the server machine.

The echo commands from the text file are executed by the shell and thus produce the file mini-ide.jar.b64. We also transfer a file README.TXT in the same manner. This file contains the commands to decode and unpack the Mini-IDE, which we simply paste to the shell to execute them.
It turns out that the copy and paste transfer mechanism sometimes is not very stable and chokes, which results in a corrupt archive file. In this case we can simply divide the contents of transfer_archive.txt by inserting some blank lines and copy everything in separate chunks.

Let's look at a quick walk through of the unpack process:

```
[oracle@localhost Mini_IDE]$ ls
mini-ide.jar.b64  README.TXT
[oracle@localhost Mini_IDE]$ cat README.TXT
Use the following commands to unpack and install.
On LINUX (tested on Linux  2.6.18-194.el5 #1 SMP Mon Mar 29 20:06:41 EDT 2010 i686 i686 i386
GNU/Linux)
================================================
base64 -i -d mini-ide.jar.b64 > mini-ide.jar
jar xvf mini-ide.jar
rm -f mini-ide.jar.b64
find . -name '*.sh' -exec dos2unix {} \;

On MAC OSX
================================================
base64 -i mini-ide.jar.b64 -D > mini-ide.jar
rm  mini-ide.jar.b64
jar xvf mini-ide.jar
```

We start with these files.

```
[oracle@localhost Mini_IDE]$ base64 -i -d mini-ide.jar.b64 > mini-ide.jar
[oracle@localhost Mini_IDE]$ ls
mini-ide.jar  mini-ide.jar.b64  README.TXT
```

Decode Base64

```
[oracle@localhost Mini_IDE]$ jar xvf mini-ide.jar
 inflated: control.sh
  created: Converter/
 inflated: Converter/ide.sh
  created: Converter/src/
  created: Converter/src/converter/
  created: Converter/src/converter/client/
 inflated: Converter/src/converter/client/ConverterClient.java
  created: Converter/src/converter/ejb/
 inflated: Converter/src/converter/ejb/Converter.java
 inflated: Converter/src/converter/ejb/ConverterBean.java
  created: Converter/src/converter/web/
 inflated: Converter/src/converter/web/ConverterServlet.java
  created: Converter/WebContent/
  created: Converter/WebContent/WEB-INF/
  created: Converter/WebContent/WEB-INF/classes/
 inflated: Converter/WebContent/WEB-INF/web.xml
 inflated: Converter/WebContent/WEB-INF/weblogic.xml
```

Unpack the archive file.

```
[oracle@localhost Mini_IDE]$ find .
.
./mini-ide.jar.b64
./Converter
./Converter/src
./Converter/src/converter
./Converter/src/converter/client
./Converter/src/converter/client/ConverterClient.java
./Converter/src/converter/web
./Converter/src/converter/web/ConverterServlet.java
./Converter/src/converter/ejb
./Converter/src/converter/ejb/ConverterBean.java
./Converter/src/converter/ejb/Converter.java
./Converter/ide.sh
./Converter/WebContent
./Converter/WebContent/WEB-INF
./Converter/WebContent/WEB-INF/classes
./Converter/WebContent/WEB-INF/web.xml
./Converter/WebContent/WEB-INF/weblogic.xml
./README.TXT
./mini-ide.jar
./control.sh
[oracle@localhost Mini_IDE]$
```

These are all files of the Mini-IDE after the unpacking of the distribution

5 WLS Setup

In order to set up the WLS Domain we first take a quick look at the file control.sh which will be used to create and control the domain.

```
#!/bin/sh
#================================================
# control.sh
# create, start, stop and conrol a wls domain.
#
# (c)2012 weblogic-corner.blogspot.de
#================================================

#--------------------set the environment
export CURR_DIR=`pwd`
export PROJECT_HOME=$CURR_DIR

#--------------------adjust your settings here
##Linux settings
export JAVA_HOME=/labs/wls1211/jdk160_29
export WLS_HOME=/labs/wls1211/wlserver_12.1
## windows settings
#export JAVA_HOME=D:\\10Oracle\\01Middleware\\jdk160_29
#export WLS_HOME=D:\\10Oracle\\01Middleware\\wlserver_12.1
## Mac Settings
#export WLS_HOME=/Users/uAries/Oracle/Middleware/wlserver_12.1
#export JAVA_HOME=/usr
```

You have to insert your path to the Java and WLS installation here. Comment the lines for the other platforms.

```
export DOMAIN_NAME=testDomain
export SERVER_NAME=testServer
export USER_NAME=weblogic
export USER_PASSWORD=welcome1
export LISTEN_PORT=9000
export LISTEN_PORT_SSL=9200
export LISTEN_ADDRESS=localhost
export ADMIN_URL="t3://$LISTEN_ADDRESS:$LISTEN_PORT"
export OUT_FILE=$CURR_DIR/$SERVER_NAME.out
```

Here we set the parameters to start the domain. You can choose your own values or just go with the defaults. If you want to use a remote client you have to enter your IP address or DNS name, instead of loalhost.

```
export WLS_JAVA_OPTIONS=
export WLS_JAVA_OPTIONS="$WLS_JAVA_OPTIONS -Dweblogic.Domain=$DOMAIN_NAME"
export WLS_JAVA_OPTIONS="$WLS_JAVA_OPTIONS -Dweblogic.Name=$SERVER_NAME"
export WLS_JAVA_OPTIONS="$WLS_JAVA_OPTIONS -Dweblogic.management.username=$USER_NAME"
export WLS_JAVA_OPTIONS="$WLS_JAVA_OPTIONS -Dweblogic.management.password=$USER_PASSWORD"
export WLS_JAVA_OPTIONS="$WLS_JAVA_OPTIONS -Dweblogic.management.GenerateDefaultConfig=true"
export WLS_JAVA_OPTIONS="$WLS_JAVA_OPTIONS -Dweblogic.ListenAddress=$LISTEN_ADDRESS"
export WLS_JAVA_OPTIONS="$WLS_JAVA_OPTIONS -Dweblogic.ListenPort=$LISTEN_PORT"
export WLS_JAVA_OPTIONS="$WLS_JAVA_OPTIONS -Dweblogic.ssl.ListenPort=$LISTEN_PORT_SSL"
```

We feed these Java options to the WLS start. Note the switch GenerateDefaultConfg=true which enables the create of the domain by just starting the server.

```
export JAVA_OPTIONS=
export JAVA_OPTIONS="$JAVA_OPTIONS -classpath $WLS_HOME/server/lib/weblogic.jar"
export JAVA_OPTIONS="$JAVA_OPTIONS -Xms512m"
export JAVA_OPTIONS="$JAVA_OPTIONS -Xmx1024m"
export JAVA_OPTIONS="$JAVA_OPTIONS -XX:MaxPermSize=256m"
# Add this line for MAC OSX
#export JAVA_OPTIONS="$JAVA_OPTIONS -Djava.endorsed.dirs=$WLS_HOME/endorsed"

#--------------------set the environment
```

Here we can add some additional Java options which are also used for calls to WLST. On OSX you may need to include the endorsed directory.

```
usage()
{
        echo "Usage: $0 { create | start | stop | status | kill | setenv} -  a WLS domain."
        echo "                create: create a simple WLS sinlge server domain in the current directory"
        echo "                start: start the server with nohup."
        echo "                stop: shutdown the server."
        echo "                status: report the status of the server process."
        echo "                kill: kill the server process."
        echo "                setenv: export the environment only. "
                exit 1
}
```

This is just the usage message which is the script default option.

```
if [ $# != 1 ]; then
        usage
fi

case "$1" in
'create')
        echo "creating the domain  ..."
        pwd
        mkdir $PROJECT_HOME/$DOMAIN_NAME
        cd    $PROJECT_HOME/$DOMAIN_NAME

        echo $WLS_JAVA_OPTIONS
        echo $JAVA_OPTIONS

        ## Starting the Domain.
        nohup $JAVA_HOME/bin/java $JAVA_OPTIONS $WLS_JAVA_OPTIONS weblogic.Server  > $OUT_FILE 2>&1 &
        echo "sending output to $OUT_FILE "
        cd $CURR_DIR
        ;;
'start')
```

We create the domain in a separate directory. We start the weblogic server with the options that we set before. The nohup command enables us to leave the server running when we logout. We redirect error and output stream into choosen output file and start the process in the background.

```
        echo "starting the server with nohup ..."
        cd    $PROJECT_HOME/$DOMAIN_NAME

        ## Starting the Domain.

        #echo "nohup $JAVA_HOME/bin/java $JAVA_OPTIONS $WLS_JAVA_OPTIONS weblogic.Server  > $OUT_FILE 2>&1 &"
        nohup $JAVA_HOME/bin/java $JAVA_OPTIONS $WLS_JAVA_OPTIONS weblogic.Server  > $OUT_FILE 2>&1 &
        cd $CURR_DIR
        echo "sending output to $OUT_FILE "

        ;;
```

The server start is basically the same command as the create.

```
'stop')
        echo "stopping the server with wlst ..."
        WLST_FILE=$CURR_DIR/$SERVER_NAME.py

        ## Creating a python stop script.
        echo "connect('${USER_NAME}' ,'${USER_PASSWORD}', url='${ADMIN_URL}', adminServerName='${SERVER_NAME}')" > $WLST_FILE
        echo "shutdown('${SERVER_NAME}','Server', ignoreSessions='true')" >> $WLST_FILE
        echo "exit()" >>  $WLST_FILE

        ## Shutting down the server
        $JAVA_HOME/bin/java $JAVA_OPTIONS $WLS_JAVA_OPTIONS weblogic.WLST $WLST_FILE  2>&1

        ## cleainng up
        rm -f $WLST_FILE

        ;;
```

We stop the server by creating a WLST script, which connects to the server and issues a shutdown command.

This is a jython script which we execute within java.

```
'status')
        echo "geting status of server  ..."
        echo "PID     USER     START   SIZE_K     SERVERNAME     DOMAIN"
        # grep the running server from ps and format the output with awk
        ## Linux (Red Hat Enterprise 5.5)
        (export UNIX95=true;ps eax -o "pid user start_time size args") | grep "${PROJECT_HOME}" | grep "Dweblogic.Name=${SERVER_NAME}"
 egrep -v "grep|start_time" | awk '{print $1 "\t" $2 "\t" $3 "\t" $4 "    " var " of " var2}' var=${SERVER_NAME}  var2=${DOMAIN_NAME}

        ##Mac OSX
        #(export UNIX95=true;ps eax -o "pid user start pmem args") | grep "com.apple.java.jvmTask" | grep "${PROJECT_HOME}" | grep
'Dweblogic.Name=${SERVER_NAME}" | egrep -v "grep|awk"   | awk '{print $1 "\t" $2 "\t" $3 "\t" $4 "    " var " of " var2}'
var=${SERVER_NAME}  var2=${DOMAIN_NAME}

                ;;
```

We use ps, grep and awk to find and print the PID and statistics of the running server.

```
'kill')
        echo "killing server  ..."
        #
        PID_FILE=${SERVER_NAME}.pid

        # grep the running server from ps, store the pid in PID_FILE
        ## Linux (Red Hat Enterprise 5.5)
        (export UNIX95=true;ps eax -o "pid user start_time size args") | grep "${PROJECT_HOME}" | grep "Dweblogic.Name=${SERVER_NAME}"
 egrep -v "grep|start_time" | awk '{print $1 }' > ${PID_FILE}
        ##Mac OSX
        #(export UNIX95=true;ps eax -o "pid user start pmem args") | grep "com.apple.java.jvmTask" | grep "${PROJECT_HOME}" | grep
'Dweblogic.Name=${SERVER_NAME}" | egrep -v "grep|awk"   |  awk '{print $1 }' > ${PID_FILE}

        echo "Really killing process ` cat ${PID_FILE}` ? (y/n)"
        read ANSWER
        if [ "$ANSWER" == "y" ]; then
                kill -9  ` cat ${PID_FILE}`;
                echo "process   ` cat ${PID_FILE}` killed!";
                exit 1;
        else
                echo "Ok, exiting with out killing.";
        fi

        # cleaning up
        rm -f ${PID_FILE}
        ;;
```

We use ps and grep to find the PID of the running server. Since every Unix is different, you may need to adjust the grep patterns. The OSX example is given in comments.

Make sure that you know what you do before killing processes. Here we prompt he user for confirmation.

```
'setenv')
        echo "Setting the environement ..."
        # nothing to do here
        ;;
```

This is used to export the environment to the shell or another script.

```
*)
        usage
        ;;
esac
```

Now let's look at a walk-through of the WLS domain creation which is really only one command to create and start it.

```
[oracle@localhost Mini_IDE]$ sh control.sh create
creating the domain  ...
/home/oracle/Mini_IDE
-Dweblogic.Domain=testDomain -Dweblogic.Name=testServer -
Dweblogic.management.username=weblogic -Dweblogic.management.password=welcome1 -
Dweblogic.management.GenerateDefaultConfig=true -Dweblogic.ListenAddress=localhost -
Dweblogic.ListenPort=9000 -Dweblogic.ssl.ListenPort=9200
-classpath /labs/wls1211/wlserver_12.1/server/lib/weblogic.jar -Xms512m -Xmx1024m -
XX:MaxPermSize=256m
sending output to /home/oracle/Mini_IDE/testServer.out
```

This is the call to create the domain. The shell points us to the ouput file.

```
[oracle@localhost Mini_IDE]$ sh control.sh status
geting status of server  ...
```

Here we can check if the server process is running.

```
PID     USER    START   SIZE K     SERVERNAME     DOMAIN
3481    oracle  09:01   1433556    testServer of testDomain
```

To stop the server we can either use the kill or the stop options. Kill sends the SIGQUIT to the process and stop connects to the server and issues a shutdown command.

6 Build and run the Converter Example

Now we are ready to build and run the converter example. Therefore we change to the Converter directory and use the script ide.sh. Let's first take a look at this script.

```
#!/bin/sh
#=================================================
# ide.sh
# operations for the mini ide
# build and deploy the enterprise application and run the client.
#
# (c)2012 weblogic-corner.blogspot.de
#=================================================

#FIND=D:/17Tools/shell.w32-ix86/find.exe
# set the environment:
. ../control.sh setenv

usage()
{
        echo "Usage: $0 { build | deploy | undeploy | run | clean } - the Converter Project"
        echo "                build: build the client and the web application"
        echo "                deploy: deploy the the web-application to the server."
        echo "                undeploy: undeploy the web-application from the server"
        echo "                run: run the client."
        echo "                clean: clean the project."
        echo "                ."
                exit 1
}


#---------clean
clean()
{
        rm -rf ./build
        rm -rf ./dist
}

#---------deploy
deploy()
{
        WLST_FILE=$CURR_DIR/$SERVER_NAME.deploy.py

        ## Creating a python stop script.
        echo "connect('${USER_NAME}' ,'${USER_PASSWORD}', url='${ADMIN_URL}', adminServerName='${SERVER_NAME}')" > $WLST_FILE
        echo "deploy('ConverterWeb', '$CURR_DIR/dist/ConverterWeb.war/', upload=true, stageMode='stage')" >> $WLST_FILE

        echo "exit()" >>  $WLST_FILE
#
        ## Calling the WLST script
        $JAVA_HOME/bin/java $JAVA_OPTIONS $WLS_JAVA_OPTIONS weblogic.WLST $WLST_FILE

        ## cleaning up
        rm -f $WLST_FILE

        ##        Give the URL for the browser
        echo ""
        echo "The ConverterServlet can be found at: http://$LISTEN_ADDRESS:$LISTEN_PORT/converter"
}
#---------undeploy
undeploy()
{
        WLST_FILE=$CURR_DIR/$SERVER_NAME.deploy.py

        ## Creating a python stop script.
        echo "connect('${USER_NAME}' ,'${USER_PASSWORD}', url='${ADMIN_URL}', adminServerName='${SERVER_NAME}')" > $WLST_FILE
        echo "undeploy('ConverterWeb')" >> $WLST_FILE

        echo "exit()" >>  $WLST_FILE

        ## Calling the WLST script
        $JAVA_HOME/bin/java $JAVA_OPTIONS $WLS_JAVA_OPTIONS weblogic.WLST $WLST_FILE

        ## cleaning up
        rm -f $WLST_FILE
}
#---------build
build()
{
        mkdir build
        echo "building the project  ..."
        $JAVA_HOME/bin/javac -sourcepath ./src -d build -cp $WLS_HOME/server/lib/weblogic.jar src/converter/ejb/*.java
src/converter/web/*.java src/converter/client/*.java
        mkdir dist
```

We source the control.sh script to export the environment into this script. We need access to java and to the running WLS server.

To deploy the war file we generate a WLST script, which connects to the running server and calls the deploy command.

We feed this script to the WLST class, running a java process.

Undeploy is also done with a WLST script.

We create a build and distribution directory for the build processes. Then we compile all java classes.

In the distribution directory we create a directory to hold the contents of the web application.

We copy the deployment descriptors and class files there.

Eventually we create the war file which is the deployment unit.

```
            mkdir dist/ConverterWebWar
            cp -r WebContent/* dist/ConverterWebWar
            cp -r build/*  dist/ConverterWebWar/WEB-INF/classes

            $JAVA_HOME/bin/jar cvf dist/ConverterWeb.war -C dist/ConverterWebWar .
            echo "Contents of ConverterWeb.war  ..."
            $JAVA_HOME/bin/jar tvf dist/ConverterWeb.war
}

#---------run
run()
{
            echo "running the client  ..."
            $JAVA_HOME/bin/java  -cp ./build:$WLS_HOME/server/lib/weblogic.jar  converter/client/ConverterClient
            ## on Windows use:
            #java -cp ./build;D:/10Oracle/01Middleware/wlserver_12.1/server/lib/weblogic.jar converter/client/ConverterClient
}
```

The client was already compiled during the build. Here we just start it with java.

```
#---------The MAIN ROUTINE STARTS HERE
if [ $# != 1 ]; then
            usage
fi

case "$1" in
'build')
            echo "building the project  ..."
            build
            ;;
'deploy')
            echo "deploying the application  ..."
            deploy
            ;;
'undeploy')
            echo "removing the application  ...."
            undeploy
            ;;
'clean')
            echo "cleaning  ..."
            clean
            ;;
'run')
            echo "run the client  ..."
            run
            ;;

*)
        usage
            ;;
esac
```

This is the main routine of the script.
We switch to the corresponding functions.

Now let's have a look at the walk-through of the build, deployment and test process.

```
[oracle@localhost Converter]$ sh ide.sh build
Setting the environement ...
building the project  ...
building the project  ...
added manifest
adding: WEB-INF/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/converter/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/converter/client/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/converter/client/ConverterClient.class(in = 2194) (out= 1212)(deflated
44%)
adding: WEB-INF/classes/converter/web/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/converter/web/ConverterServlet.class(in = 2667) (out= 1439)(deflated
46%)
adding: WEB-INF/classes/converter/ejb/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/converter/ejb/Converter.class(in = 265) (out= 202)(deflated 23%)
adding: WEB-INF/classes/converter/ejb/ConverterBean.class(in = 775) (out= 474)(deflated 38%)
adding: WEB-INF/web.xml(in = 612) (out= 256)(deflated 58%)
adding: WEB-INF/weblogic.xml(in = 543) (out= 238)(deflated 56%)
Contents of ConverterWeb.war  ...
     0 Tue Jul 03 09:28:48 PDT 2012 META-INF/
    71 Tue Jul 03 09:28:48 PDT 2012 META-INF/MANIFEST.MF
     0 Tue Jul 03 09:28:48 PDT 2012 WEB-INF/
     0 Tue Jul 03 09:28:48 PDT 2012 WEB-INF/classes/
     0 Tue Jul 03 09:28:48 PDT 2012 WEB-INF/classes/converter/
     0 Tue Jul 03 09:28:48 PDT 2012 WEB-INF/classes/converter/client/
  2194 Tue Jul 03 09:28:48 PDT 2012 WEB-INF/classes/converter/client/ConverterClient.class
     0 Tue Jul 03 09:28:48 PDT 2012 WEB-INF/classes/converter/web/
  2667 Tue Jul 03 09:28:48 PDT 2012 WEB-INF/classes/converter/web/ConverterServlet.class
     0 Tue Jul 03 09:28:48 PDT 2012 WEB-INF/classes/converter/ejb/
   265 Tue Jul 03 09:28:48 PDT 2012 WEB-INF/classes/converter/ejb/Converter.class
   775 Tue Jul 03 09:28:48 PDT 2012 WEB-INF/classes/converter/ejb/ConverterBean.class
   612 Tue Jul 03 09:28:48 PDT 2012 WEB-INF/web.xml
   543 Tue Jul 03 09:28:48 PDT 2012 WEB-INF/weblogic.xml
 [oracle@localhost Converter]$ sh ide.sh deploy
```

We start the build process here.

We print out the contents of the war file for information.

We start the deployment process here.

```
Setting the environement ...
deploying the application  ...

Initializing WebLogic Scripting Tool (WLST) ...

Welcome to WebLogic Server Administration Scripting Shell

Type help() for help on available commands

Connecting to t3://localhost:9000 with userid weblogic ...
```

The generated Script connects to the running server.

```
Successfully connected to Admin Server 'testServer' that belongs to domain 'testDomain'.

Warning: An insecure protocol was used to connect to the
server. To ensure on-the-wire security, the SSL port or
Admin port should be used instead.

Deploying application from /home/oracle/Mini_IDE/Converter/dist/ConverterWeb.war to targets
(upload=false) ...
<Jul 3, 2012 9:29:17 AM PDT> <Info> <J2EE Deployment SPI> <BEA-260121> <Initiating deploy
operation for application, ConverterWeb [archive:
/home/oracle/Mini IDE/Converter/dist/ConverterWeb.war], to testServer .>
.Completed the deployment of Application with status completed
Current Status of your Deployment:
Deployment command type: deploy
Deployment State       : completed
Deployment Message     : [Deployer:149194]Operation "deploy" on application "ConverterWeb" has
succeeded on "testServer".


Exiting WebLogic Scripting Tool.

<Jul 3, 2012 9:29:20 AM PDT> <Warning> <JNDI> <BEA-050001> <WLContext.close() was called in a
different thread than the one in which it was created.>

The ConverterServlet can be found at: http://localhost:9000/converter
[oracle@localhost Converter]$
[oracle@localhost Converter]$ sh ide.sh run
Setting the environement ...
run the client  ...
running the client  ...
Amount: 2000
Yen: 166120.40
Euro: 1545.19
[oracle@localhost Converter]$
```

WLST is telling us, which file it is deploying.

The deployment was successful.

We test the deployment by running the client.

If we have a browser running on the remote server we can also test our servlet by pointing it to the address http://localhost:9000/converter.

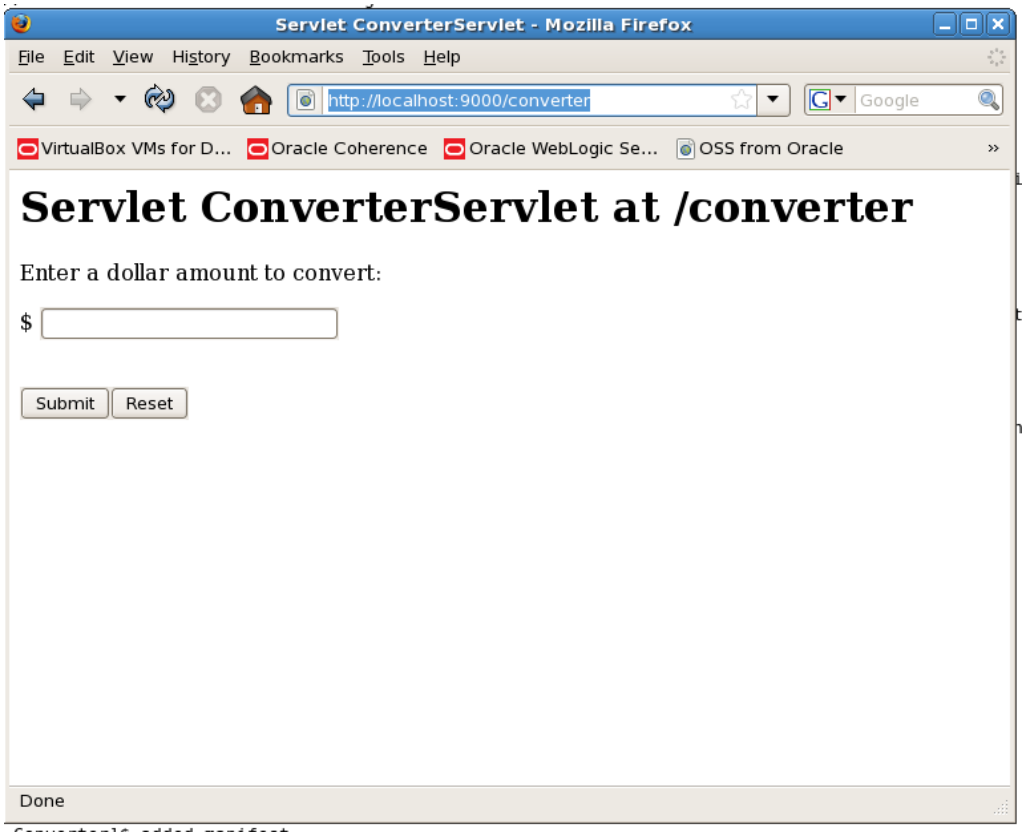

Figure 4. Test of the ConverterServlet.

The Converter example was taken from the Java EE6 Tutorial (http://docs.oracle.com/javaee/6/tutorial/doc/gipss.html). It is slightly modified to run on WLS. I have the added the remote java client.

7 Links

The transfer Text file:
https://dl.dropbox.com/u/16989587/weblogic-corner/Mini_IDE/transfer_archive.txt

The base64 encode jar archive:
https://dl.dropbox.com/u/16989587/weblogic-corner/Mini_IDE/mini-ide.jar.b64

The jar archive:
https://dl.dropbox.com/u/16989587/weblogic-corner/Mini_IDE/mini-ide.jar

This document as PDF:
https://dl.dropbox.com/u/16989587/weblogic-corner/Mini_IDE/Mini_IDE.pdf

8 Attachment: transfer-archive.txt

We attach the file transfer-archive.txt for convenience.

```
echo "" > mini-ide.jar.b64
echo "UEsDBBQACAgIABpo40AAAAAAAAAAAAAAAAAKAAQAY29udHJvbC5zaP7KAADdWG1P40YQ/mz/ijnH" >> mini-ide.jar.b64
echo "aqACJ05LJYLcKiK0pQWCYuidWqrcnr0kBttr7doESvnvnV17HeeFK/fSSm2EcLyzOzP7zOzMs2m9" >> mini-ide.jar.b64
echo "6ryL0o6YmS3vgz9mCwKW5pzFjlQAAackpzsgcsJz+WAZkDSUk3AOEJjHAkKWkCh1zBYu2Aq2e123" >> mini-ide.jar.b64
echo "B3P6LmbTKNgNGE8pd+SbyFjuhPSj/DJbuxs+guaQzyjQ9C7iLE1ompv0PmM8h8PL8XgyPB57b7N5" >> mini-ide.jar.b64
echo "+FaPno9HPx0dXkx+HJ0eebae84x2clOIHB5YwQEN5VE6FTCjHDfQOonS4r4e1dp/GvwyKFV3YvJO" >> mini-ide.jar.b64
echo "dBAct+e6nZvw1v2mO+nt64mvT/xN8/BJ+R3lE7fnuGgF5lEasrlYGGqtWRr2r67c7oiTIKZXV133" >> mini-ide.jar.b64
echo "NArDmM4Jx7eF3daq4fctW3PjlATgr7qw2MMlThedYsAjKjqlys5C3+qu1rEqBDflp5IMR6eD47PJ" >> mini-ide.jar.b64
echo "2QBlORX5UKWXlvpH41+Oxgupr3Rr6aWvZToBlyTnA99/PRoPURoHLKGulp4c+xdHZ5Pz0fjC2+92" >> mini-ide.jar.b64
echo "uxvGJ75/4u331mSD4XB85PtezAISz5ioM3AwPMVtXI5PPCv/qt/p2MsL+nZDuaUXjS4vJt8fnzSS" >> mini-ide.jar.b64
echo "s2M3tuywIjebWaRgHJ1fHI/OfO9ZgWWvDsHuUCPklAB7dgN46yNVnZGEek2HP1ZRQlIypfJAOwWm" >> mini-ide.jar.b64
echo "T6r01tH9DFozIsSc8bDSqjPjM2j+gWLBw6o5pNekiPNDll5HUy/nBf1Y5SeRyGk6CENOhfBW8ujT" >> mini-ide.jar.b64
echo "lJ7jSm9jJn6gOiHi51TKk2MtzvfGlF2xtGwliDFYGclnYOuq0ylrSieO3nVqH24It2D98yIbbxKx" >> mini-ide.jar.b64
echo "5/aST1h/73Z7X3+Cgjf9U3J/TnniR39Qr7f3TakLmypGHttcJCCOUgrXjMPp4BBG/pvlavqc5uEN" >> mini-ide.jar.b64
echo "uSMOxVbCBQ2dMOKYRDWQehyNmRt8r/fw4gZsmoXAs7C1bT6aBg1mDKxLOdAHuwuPFaWAP0tOoZ5I" >> mini-ide.jar.b64
echo "KtRrXgj8chvFsXynmEt3T7ALSDPQW00zLNBaK8+MUmNfayYgoiSLqVokojSeUiizpVIB+CedDgrO" >> mini-ide.jar.b64
echo "0WFAQGiQM/6wrlr52K9clWsqRfMIkzFlsyLb4I/cEK6ZFTk277SxbONcues+cKriqOaWQLDrpsGM" >> mini-ide.jar.b64
echo "swAP/wYNEq5+CdqL5pe49qHKnJXoAUvjBwcs0zDofZSDaz5hQKNr+A3sFrzywIXfD+Si1DRUmM3r" >> mini-ide.jar.b64
echo "yDQDIihYtmshtGa7jEN7W1tW78gdlK0qBOA4DhpBbmYayS1GAOwmN+s0m5FpBKF0/T0zKlNrFUqP" >> mini-ide.jar.b64
echo "L42ZBnIaX4ZUOzWsGKyhQlpNV1YklZbHZ1nFuqGa8DolIQH4FmzdzKH37RcufKHxEHjkpGXs5VmB" >> mini-ide.jar.b64
echo "EWCNmZba7IKZGgcHZlul3wJP0XR9LSEVsoZpvAi154BASWnrHwXE+scBX8byRfBLyA2JOcuakLMs" >> mini-ide.jar.b64
echo "2wQ5kttcI44uvp+8ZQ8l5If6QBDIHvIZS8saKAIeZblTHxuWpliXttr2Y018ntqwo981ZXlq70DB" >> mini-ide.jar.b64
echo "Yw/Ha8opx0iYRGkJjqJlKG8489TetiRktdP1Vqu6tbU6f6ddKkPV0TRlnPpYYCKWCq8tyY3St0mh" >> mini-ide.jar.b64
echo "rCNbpbApLZMPjSkkViqlaXxSSkgrDVtQ5oOMLdrE6wnmONrExDMNnsDu9ZLXdQLIMrxIgSlVji6q" >> mini-ide.jar.b64
echo "c5UGOvrlrPPjoSqzMkLy6V8Mxhfyefzr0eRnJSpBlZiq1/JAYuK1YIpdQIHAizRVxkoT15wlkAl1" >> mini-ide.jar.b64
echo "88bun5CybFcJrPKQzG/V5sqL6daYhvAjTjtKc8ozHmF53nP2cC9b+jp0dvxmf0+x0gPUTMk97KL7" >> mini-ide.jar.b64
echo "WRSC5Ntlx5vkUYIRQToChE+FtY1tWflo2Y/NqvJk1YLVq8BSCslpVM3bvUNA8cufCztSiLuA9iP6" >> mini-ide.jar.b64
echo "i43IdsG6yi2we9Xzq+r5ddnLLLgjHP9jJNTXHp4NfKyYBCXCwUbde1J5IC+1kkIZrQ/CBLkEJqtw" >> mini-ide.jar.b64
echo "4F3SIRlyDkdRrZu75IKIW+vzw4UAyc3/i0ghEVDHQRKMxWGQb40Erbp5C88THoCqBi5pdxBFVPWC" >> mini-ide.jar.b64
echo "LFe/MnGqZkjokSxonf+/DMdIYMl81Bt8+u8n5t/vUOfQmJI4fgCdShVjhbcQYFgbK97Cd7D10Em3" >> mini-ide.jar.b64
echo "McOwcYYwOPNfy+pqGoqXWnY5YIHngYUkviaoihvD7j5sUHogGa7yQxveNEt5R8NX1kFNiPEbjQWt" >> mini-ide.jar.b64
echo "l49ud0BK5B5UKcayrDflyHVIkFXey76j8r3Zd5rAlG1H0fPFSat+d2vydPUzQ33kUoZXQzmBYRMt" >> mini-ide.jar.b64
echo "f59Umswvt+v7XMnUJZekggTmX1BLBwjFdtu7zgYAACMWAABQSwMECgAACAAARwHiQAAAAAAAAAAA" >> mini-ide.jar.b64
echo "AAAAAAoAAABDb252ZXJ0ZXIvUEsDBBQACAgIADaI40AAAAAAAAAAAAAAAAAQAAAAQ29udmVydGVy" >> mini-ide.jar.b64
echo "L2lkZS5zaN1WW2/iRhR+9vyKs4BEsortkEpt5YhKNDi7VAEimywPVYWMPYRJjccaG5IoyX/vmYvB" >> mini-ide.jar.b64
echo "LKTb7VNVSyFzOdfvnPlmmh/cOcvcYkma3e/+SBNYQh2pDDynIioZzwpYcAHlksKKZUwK4O58zdIE" >> mini-ide.jar.b64
echo "oiyBhOYpf1b7NCupyAUrKER5nrJY6Sspsc6USJwylHJIE22cxKcX550LeKTzlN+z2I65yKhw5KzI" >> mini-ide.jar.b64
echo "eelIT/8iCdK8Hoz63b7ndn6acJ4WiAZNU+fxhwubPf38o7tgWeLQJ5lHQUsT+oYJnq0wOI844Dhu" >> mini-ide.jar.b64
echo "zLNS8BTBkEK4T8i6iO7pySl5IRaNlxwad3LBg9Y5vBhEXis8XmGdbYcy+1fMnUYZvIGtPF7xbEMF" >> mini-ide.jar.b64
echo "Aga3gj/QuGxAZRb0ZymTnrG8Q08BKqcIXB3oQwM6AK9eI6Nn1wtUcrVcUIEROYdmqky8XU7HzCwE" >> mini-ide.jar.b64
echo "X9UMHdpBGLyvO+FQSsHkGbSkZK7xOSLqNIh10SdWQoe8ESy8XX1Km6hfXTCxAlsswHEVmrV5worS" >> mini-ide.jar.b64
echo "kto7ZZ0j0f+0+vQmnMyuBzd+t3V1FwSz/iBwW6EffPGD2ag39B0t7OTPhFjNJlwJiqBk9xBB/lwu" >> mini-ide.jar.b64
echo "EZ2i5DkUsWA5tr/JA3ssw8xO2q2Xu9BYemvDWTW/7YXhdBz039pnsBZpF9d7/eFgNLsLbuRalOCR" >> mini-ide.jar.b64
echo "DBXao2hF5X4tprf2aQN+gdY29sqtSay9bcEpnaO19i41CYpb33YeI+HKKFAzSrqlWNMzTAnbf8gT" >> mini-ide.jar.b64
echo "dKyGyt+ew8qjrNGJ3q1vNzVWUZpKqGSt5Z5BiVit33pferPP46GvOO0h2kSg18a3k8F4FCpbs72V" >> mini-ide.jar.b64
echo "ikwcZWnnC3tHOVMtIb2tc9BdsNhDSApZn9hGnxTEect+c8EfsbOrlBrVYFI/zLIYKXJKjL07p6iL" >> mini-ide.jar.b64
echo "Jwai0oNlWeae67ZuBuHEH816/X7gh6FXzW/HwURSjrbSwH7ctWN16Eg1+B+25Da1/ab83ob6r/WT" >> mini-ide.jar.b64
echo "tVdITT3qV5dw9WfCBBhG0ompSRW9YT7A+0hS3ZH4Y7ALvhYxzaNyiWxWCFxJzJVhx7nOSKloVnZT" >> mini-ide.jar.b64
echo "Nne3ST1EAlBl13kufZi7Hx0Fzf4G6hzf0ERe7VVJSQapj/fYZBrhMcLgbAE4w51SWfhbQZUSyhwV" >> mini-ide.jar.b64
echo "cqf+r/ZgdI3BREVBC3IELAHxZnGoLZkN7KujZmF7LEyMBfAFHGi/Vx7kjfc8Wl9dOXgrEvyrPy5w" >> mini-ide.jar.b64
echo "mlWNYC7+9/sAVLHN7eZ9u+hwUL9thFdqrpobOWKKbyXkPVgX1MM15avm6lK+ss7HIsJz4J53hixJ" >> mini-ide.jar.b64
echo "Uor5Ufcx1b5nnQun824c3wxj/1qXXDvsDUYQjO8mg5EP4aQXTEL47Ac+YQv4HVpN+NCFDvxxKYHL" >> mini-ide.jar.b64
echo "iKUebWTBCIkjfJg2Wp0GsIy0Vfjt039y7swBvbzEqrU1T+0U9bzSrL+HjLYWsCr9iul2FgRd8c17" >> mini-ide.jar.b64
echo "BqSF7QVgTCjO2elvKcg41G8fI4tdVPO09/aq5GXvaWny8ZSYtxVo3HCd0CKKyV9QSwcIG84aimME" >> mini-ide.jar.b64
echo "AABYDAAAUEsDBAoAAAgAAC1i3kAAAAAAAAAAAAAAAAAOAAAAQ29udmVydGVyL3NyYy9QSwMECgAA" >> mini-ide.jar.b64
echo "CAAAO6LhQAAAAAAAAAAAAAAAAAAgAAABDb252ZXJ0ZXIvc3JjL2NvbnZlcnRlci9QSwMECgAACAAA" >> mini-ide.jar.b64
echo "O6LhQAAAAAAAAAAAAAAAAAAAAAAB8AAABDb252ZXJ0ZXIvc3JjL2NvbnZlcnRlci9jbGllbnQvUEsDBBQA" >> mini-ide.jar.b64
echo "CAgIAPKr4UAAAAAAAAAAAAAAAAAAAAAzAAAAQ29udmVydGVyL3NyYy9jb252ZXJ0ZXIvY2xpZW50L0Nv" >> mini-ide.jar.b64
echo "bnZlcnRlckNsaWVudC5qYXZhjVbbbuM2EH12gPzDQE92NpWD9qlJu9jE8W5dGPEiUZsuFkVAy7MW" >> mini-ide.jar.b64
echo "E4p0KcpOUOTfO7zIukTphg+2LJ6ZOTM8M/SGpQ9sjZAquUVtUMep4CjN2eHB4QHPN0obuGdbFufM" >> mini-ide.jar.b64
echo "ZPEFX19iynMmztq7peEi/o0VmWFLgX2bM/LNjNJ9e3NemPb7xxjvl/H094vua8lyLtfxREmDjy+M" >> mini-ide.jar.b64
echo "wu5McsOZ+H/QlfuaPqa4MVxJl/AeWZfD8phUvxxoUy4FTyEVrChgvzVxZYN/Dw+A1nj8gcj7543m" >> mini-ide.jar.b64
echo "W2YQCsMM2e0t6iBnbeCN0cQMSi3gV4jMT6fjsVApE5kqzOnPJycnUWXgmXQ4DL3917+B6XUxqig9" >> mini-ide.jar.b64
echo "B2ZHR/4BjuDDhmmWOxiYzIogz5lcgeAS7dsyJ3dFBR+3goZstoqvIGdcvhJ10C2QlxclJnH3grkz" >> mini-ide.jar.b64
echo "DcnZ5cHxSiVYmGG1EzJp0XE8KlwVvEYNxmNYowHulWFLb6UBRrnEvSSgQL3lKQYTo5+so8FgEIQE" >> mini-ide.jar.b64
echo "qXkk4uSnLTBHjGD78yTQcJ/ayJrFQqmHcjOMrARP10ItmahldYsNjV0gk1HwaD+eIWUmzWDYESzg" >> mini-ide.jar.b64
echo "KNCj3JLF5QLOS6N+WKO0jYarYLYk6Tw42M1TYTCPVWliUpo0Qg6jqdZKwy7jAm1exlbBFiQkCCHD" >> mini-ide.jar.b64
echo "uCKE3vTG0NhINEsx5P5c1Tlol+WqdKcc/djQq++MSvcuUADSOTCopwt80yp3+xr/KelMwSkVqTy1" >> mini-ide.jar.b64
echo "pwZ8FfRUvxp6xyPXsc3YTAjnuFVvUpgQTCfqC0qSDwXK1MqSsppp0OQSCNFL4QnleZV0PT2afldv" >> mini-ide.jar.b64
echo "JUOuEjUttfoOFQspeskg7fSwqR3v2Ta7rU8gHnUKEbwD9iYLStXD3xzDMvImNe+6Vs/Vw6Caj1U7" >> mini-ide.jar.b64
echo "9jQiVUirXQHdXmkMBLtCa0Nr0YF8ohozCW2vXdz+mvvFi/3Yf72HLKjwNcCwWQi/snhTmmHVZbOr" >> mini-ide.jar.b64
echo "WTI7n99NFlfJ9K/k7uP5JFlcfznuWgFEO6TGXvM0vpcrHt/O25Q/spSu2qfoewE/Xy/+nF1Or+/+" >> mini-ide.jar.b64
echo "uJ4f2wvnpYFGU2rp8uoUO+uCX51UEkfdclfTCLWuhXCLsEONUEpbvUr0zCpYYupchYFNE3Nu03cj" >> mini-ide.jar.b64
echo "mwYuMxC96+HfG+azQFYg3VoPdCmXFM9kzPdW8MYL0KWU9l9CTwmdxCinbvbNn/srih7+A1BLBwiP" >> mini-ide.jar.b64
echo "5M/xeQMAAGMJAABQSwMECgAACAAAUKfhQAAAAAAAAAAAAAAAAAAwAAABDb252ZXJ0ZXIvc3JjL2Nv" >> mini-ide.jar.b64
echo "bnZlcnRlci9lamIvUEsDBBQACAgIAFyo4UAAAAAAAAAAAAAAAAAqAAAAQ29udmVydGVyL3NyYy9j" >> mini-ide.jar.b64
echo "b252ZXJ0ZXIvZWpiL0NvbnZlcnRlci5qYXZhbY07DsIwEERrLPkOW4bGF3AT8bkASkO5MUtwsL2W" >> mini-ide.jar.b64
echo "cSJQxN1J+AhFIG315s1sRHPGhsBw6CllSoraWkshhfWRU4YWe7xOUO3IcyY9S5THfFIr22zIWI/u" >> mini-ide.jar.b64
```

```
echo "2SxfphSxq501YMO4e0RDsP58gUGKxXhv4zsAB3YOU8V7CsUPviz139KNQsXbLnExh5N+l+IBUEsH" >> mini-ide.jar.b64
echo "CKLI+NCNAAAA5QAAAFBLAwQUAAgICAACquFAAAAAAAAAAAAAAAAALgAAAENvbnZlcnRlci9zcmMv" >> mini-ide.jar.b64
echo "Y29udmVydGVyL2VqYi9Db252ZXJ0ZXJCZWFuLmphdmGtUs1O3DAQviPxDiNOYbVywiIhyqoSBXqr" >> mini-ide.jar.b64
echo "SgXLgRNyksnG4NiRPckSVX33jpNsNyu04tKJD/HMfD8zSTw7PoIZ3Nq6c2pdEiySswXcO5lpBGny" >> mini-ide.jar.b64
echo "2DpQ5EEWhdJKEnrRA75pDT3Ag0OPrsVcADzbBirZgbEElc1V0c2h8TjvIQ5rZ/Mmwzkwaa48OZU2" >> mini-ide.jar.b64
echo "hECl8uBtQRvpEPA9w5pAmR6U2apmXZMhbBSVwM0MQFd5sEV/+aEyNJ7N0lWPKInqqzjOsUVta3Re" >> mini-ide.jar.b64
echo "+MYI5on10Bmn6N5QY/cyJkRJlQ7Y+PgoPLXM3uQaWdu06FhM4Gu6DBVV1dYRvMpWikpSKW7U+g4z" >> mini-ide.jar.b64
echo "VUm93Ku+B4h4JF6YRu+XA3E8G7a9CgPzCfZTlAYyLb2HgtcSUrdb3ZtQQ8NvtVN+6B3Wfy0bKsOn" >> mini-ide.jar.b64
echo "kWY0fv1PjAdoUh5tZN1nY48aK+acFH4zBQdrtEwBu5kAG2cfQu4rGNxMKtFJIpLky3lydnFyujyI" >> mini-ide.jar.b64
echo "79AcgF+ei+QiWfTgET6YnqBzq7V0K/uMJvqQ9qcw+g4xKfPv2GhiybFPVHxVte6i0c1OMoRDapwZ" >> mini-ide.jar.b64
echo "UcIjPWZSY7SYTzjFw/3Tz7uXp1/bUf8cNs0iK/ud9xbtJz+1yz07q9vF/x+vfP4CUEsHCJmiGWjY" >> mini-ide.jar.b64
echo "AQAA6AMAAFBLAwQKAAAIAADkXd5AAAAAAAAAAAAAAAAAHAAAAENvbnZlcnRlci9zcmMvY29udmVy" >> mini-ide.jar.b64
echo "dGVyL3dlYi9QSwMEFAAICAgAgqvhQAAAAAAAAAAAAAAAADEAAABDb252ZXJ0ZXIvc3JjL2NvbnZl" >> mini-ide.jar.b64
echo "cnRlci93ZWIvQ29udmVydGVyU2Vydmx1dC5qYXZh5Vhta9tIEP5e6H+YE0exk1RqCwdH45i0vVzb" >> mini-ide.jar.b64
echo "4yCm8VEKhWMtjextV7vq7iqpr/S/34y0shTZTvryoXCnQNDuvD07bztycnD3DhzAM1OurVyuPDx6" >> mini-ide.jar.b64
echo "8PARnFuRKgShs8RYkN6ByHOppPDo4lrgiVJQCziw6NBeYhYDvDEVFGIN2ngoTCbz9RFUDo9qEYul" >> mini-ide.jar.b64
echo "NVmV4hGQ0kw6b+Wi8gh+JR04k/srYRHwY4qlB6lrodQUJdnVKcKV9CsgZhJAWzgweb34U6aoHYH1" >> mini-ide.jar.b64
echo "j2uJlffl4yTJ8BKVKdG62FU6Jj2JajiTBdr3qHD9d9iIV75QLJvcvcN/pUjfiyWSbX2JlozFV7g4" >> mini-ide.jar.b64
echo "ZoosSmM9vBOXIpYmfnl+VoOVRh9vEWdWav/aSpIfEAvhV/FTufwNU1kIdZ36McZ3i/jsj6fDbXay" >> mini-ide.jar.b64
echo "Qh8LTd4VbDN+jYuLZncfcyDvgdnxsdPiF/TvFn1Dvlf4oUL3FeyuNOTxjr/zMR/7Wbs6biKRJKfd" >> mini-ide.jar.b64
echo "Ge/e6S1GlVUz4YlVu5Mo2QgGcjSmMFYLCjCkSjgHQwZKM486c9BDB58oCeg5Je83bxupDmeNjGnJ" >> mini-ide.jar.b64
echo "wUHzQjk3syZF55CLoXaIg5xyfGEoY1/M5zOYpCbD6fOz+SSp37i0wubs/GKzW6BfmawusUbxaSms" >> mini-ide.jar.b64
echo "KFqtEPzarrfZGvf2+JqNjtGvrLlyMMwLkDmIVuy+Kykzc/IdWkvHMGlaWbelo5f9tbiGl8n5TpGk" >> mini-ide.jar.b64
echo "eaHy95h6zODSyIyX7LSQQqPATM92frUHPtrDFI7dHncMe455dA30p05bK0mZ6ynolBp+vi5xFHnK" >> mini-ide.jar.b64
echo "k4T7w3G6EpaoJ3/Nf7//azTe5AE/vVoHU3k46RQuMRBGLNIKJAmcV74kVm5hxFwp7zoy6YhL1qn0" >> mini-ide.jar.b64
echo "KJqweVBCL0/eRqjfRtOor2rAiyK7ie6lVzhtE35YFZOkod+gILnNwsJk6/30brd9CPPDvYCoq0ME" >> mini-ide.jar.b64
echo "h2382Zt1eD56Kv7VaLyt7hAY48PpIELervvx5ueC7h+9BFGYSjch29iYcTkhBy1qyANt/FDKj0ZB" >> mini-ide.jar.b64
echo "+KcT0JVSY7h3D8JerFAvGSJM4cF4PDQekiC0lToLgi5vqBK72wFya4qQJU0tlC24bY09sYxOpPGq" >> mini-ide.jar.b64
echo "txWAbZ+kxSLoTmdDmyA8RaHjzCgl7Ny8QU2HaToUgyQ39WDv1Cg1kNSNMNeon7QB6O6Cvs3sawGT" >> mini-ide.jar.b64
echo "yrk5q6z5Brgs5m4EjMSxA3FndHOi3bhvLob2iSbllPM+5AQlNTQ+oUmMZqRov+Rh59LYm5kSUjeJ" >> mini-ide.jar.b64
echo "Tt5gNUSNJ0l5vUK/Ddythm7H2nlzpw526G60nwEVdfwdVXW9G5XTM81dWQT/9aosBO/xF3kjmtCF" >> mini-ide.jar.b64
echo "XoR8oj5M6TRsxF/vxJ9hInV9CXDXJa2NK95G4OnyoTU3OlppqndaiZbq5D+8fvQLQfhC9AubfB/a" >> mini-ide.jar.b64
echo "ALTB5apFIRnJpVAVb1yEjemN0R5o4Q+HvpJXzfpLjpNwNHZkRbf8DLnU1CC22v5A0fZ1tYOJb+Dd" >> mini-ide.jar.b64
echo "TKkyDq9d7QHD582cmMAEM+mNvZ8blUGGueDbnmZ4PIlSzsrSYUYFji49ifrzaDsOwjMaZN8DjSzc" >> mini-ide.jar.b64
echo "wQ4p/kvdrhTmdTaziXqDJ8k4mm4PqS9o5lQ0ojLT7qG0sfcfnT5Pz6narcxw5zCamef4g2fQwTzc" >> mini-ide.jar.b64
echo "mus0Hg8z6/bgbn9c/E+jOzM/+hPju8L7Cn1FX7rsxxV/NnOvsLKxEX4Hab+5O1/aWohkwqRLF56n" >> mini-ide.jar.b64
echo "G5Zf2yD29Nzi0OZjOmii2y+c+6XOec669ilVW40uhkCj9oB1S0x6PZGaFZ36X1BLBwhjPAOYLwUA" >> mini-ide.jar.b64
echo "AIsSAABQSwMECgAACAAAEWXeQAAAAAAAAAAAAAAABUAAABDb252ZXJ0ZXIvV2ViQ29udGVudC9Q" >> mini-ide.jar.b64
echo "SwMECgAACAAAApHhQAAAAAAAAAAAAAAAB0AAABDb252ZXJ0ZXIvV2ViQ29udGVudC9XRUItSU5G" >> mini-ide.jar.b64
echo "L1BLAwQKAAAIAABcqMZAAAAAAAAAAAAAAAAAJQAAAENvbnZlcnRlci9XZWJDb250ZW50L1dFQi1J" >> mini-ide.jar.b64
echo "TkYvY2xhc3Nlcy9QSwMEFAAICAgAYo3hQAAAAAAAAAAAAAAACQAAABDb252ZXJ0ZXIvV2ViQ29u" >> mini-ide.jar.b64
echo "dGVudC9XRUItSU5GL3dlYi54bWylUcFKw0AQvQv+w7JHITtpgyBhkx4ET/VUBW9l3S5tTJkNmW2S" >> mini-ide.jar.b64
echo "z3dITEwFseDxzbw37+1bvemrUrSuocJjJlcqlsKh9YcCj5l8fXmKHuQmv73RnXuPTF0LpiOlPRWZ" >> mini-ide.jar.b64
echo "PIVQpwBd16kuUb45wjqOV/D2vN3Zk6tMVCAFg9bJUTUrPkxrFJ1RWV8BrwBpmLmJmbLbFWz4CrVf" >> mini-ide.jar.b64
echo "7+9VTwdWU5HSYL711oThTX+eEdcbJft4NJobS7gx7kcITa5pSxcG8A0jNJXLHz2yIrhmN041XKx/" >> mini-ide.jar.b64
echo "SGxpiHI7aRS7q98vjOwhAiwzzOcqTs7f+Y9g56aMahOYhjncaVjiC+OFl55Kyz8BUEsHCEOHIDsA" >> mini-ide.jar.b64
echo "AQAAZAIAAFBLAwQUAAgICABWjeFAAAAAAAAAAAAAAAAKQAAAENvbnZlcnRlci9XZWJDb250ZW50" >> mini-ide.jar.b64
echo "L1dFQi1JTkYvd2VibG9naWMueG1slZHBTsMwDIbvSLxDlHvipoCEqra7cdpOgMRtClnUBaVJVWdL" >> mini-ide.jar.b64
echo "Hx9TCmIwCS2n5Lc/2/Ffr6bes6Md0cXQcCULzmwwcedC1/Dnpwdxz1ft9VWdPVbZvvrYOSPoIvQw" >> mini-ide.jar.b64
echo "MGIDyR4bvk9pqABmRcZRG2+liT18MfAb5gs9ofumc84y3xDeQVkUCl4260ezt70WLmDSwVii0FU4" >> mini-ide.jar.b64
echo "i+todJrHXvA3fdQSD2FuTNUh4KxZy/7NgGWsbbm9kxPu2MUfupwAJW//iB/NOW2c0Tnd+uJSq0qp" >> mini-ide.jar.b64
echo "pKrhbPAHaGJIdkpijDG1QC/KSXb8BE+C5C+cM7h9B1BLBwgMX6cv7gAAAB8CAABQSwECFAAUAAgI" >> mini-ide.jar.b64
echo "CAAaaONAxXbbu84GAAAjFgAACgAEAAAAAAAAAAAAAAAAAAAY29udHJvbC5zaP7KAABQSwECCgAK" >> mini-ide.jar.b64
echo "AAAIAABHAeJAAAAAAAAAAAAAAAACgAAAAAAAAAAAAAAAAKBwAAQ29udmVydGVyL1BLAQIUABQA" >> mini-ide.jar.b64
echo "CAgIADaI40AbzhqKYwQAAFgMAAAQAAAAAAAAAAAAAAAADIHAABDb252ZXJ0ZXIvaWRlLnNoUEsB" >> mini-ide.jar.b64
echo "AgoACgAACAAALWLeQAAAAAAAAAAAAAAAA4AAAAAAAAAAAAAAAA0wsAAENvbnZlcnRlci9zcmMv" >> mini-ide.jar.b64
echo "UEsBAgoACgAACAAAO6LhQAAAAAAAAAAAAAAABgAAAAAAAAAAAAAAAA/wsAAENvbnZlcnRlci9z" >> mini-ide.jar.b64
echo "cmMvY29udmVydGVyL1BLAQIKAAoAAAgAADui4UAAAAAAAAAAAAAAAAfAAAAAAAAAAAAAAAADUM" >> mini-ide.jar.b64
echo "AABDb252ZXJ0ZXIvc3JjL2NvbnZlcnRlci9jbGllbnQvUEsBAhQAFAAICAgA8qvhQI/kz/F5AwAA" >> mini-ide.jar.b64
echo "YwkAADMAAAAAAAAAAAAAAAAAcgwAAENvbnZlcnRlci9zcmMvY29udmVydGVyL2NsaWVudC9Db252" >> mini-ide.jar.b64
echo "ZXJ0ZXJDbGllbnQamF2YVBLAQIKAAoAAAgAAFCn4UAAAAAAAAAAAAAAAAcAAAAAAAAAAAAAAA" >> mini-ide.jar.b64
echo "AEwQAABDb252ZXJ0ZXIvc3JjL2NvbnZlcnRlci9lamIvUEsBAhQAFAAICAgAXKjhQKLI+NCNAAAA" >> mini-ide.jar.b64
echo "5QAAACoAAAAAAAAAAAAAAAAAhhAAAENvbnZlcnRlci9zcmMvY29udmVydGVyL2VqYi9Db252ZXJ0" >> mini-ide.jar.b64
echo "ZXIuamF2YVBLAQIUABQACAgIAAKq4UCZohlo2AEAAOgDAAAuAAAAAAAAAAAAAAAAAGsRAABDb252" >> mini-ide.jar.b64
echo "ZXJ0ZXIvc3JjL2NvbnZlcnRlci9lamIvQ29udmVydGVyQmVhbi5qYXZhUEsBAgoACgAACAAA5F3e" >> mini-ide.jar.b64
echo "QAAAAAAAAAAAAAAABwAAAAAAAAAAAAAAAAAnxMAAENvbnZlcnRlci9zcmMvY29udmVydGVyL3dl" >> mini-ide.jar.b64
echo "Yi9QSwECFAAUAAgICACCq+FAYzwDmC8FAACLEgAAMQAAAAAAAAAAAAAAADZEwAAQ29udmVydGVy" >> mini-ide.jar.b64
echo "L3NyYy9jb252ZXJ0ZXIvd2ViL0NvbnZlcnRlclNlcnZsZXQuamF2YVBLAQIKAAoAAAgAABFl3kAA" >> mini-ide.jar.b64
echo "AAAAAAAAAAAAAAAVAAAAAAAAAAAAAAAAAAAGcZAABDb252ZXJ0ZXIvV2ViQ29udGVudC9QSwECCgAK" >> mini-ide.jar.b64
echo "AAAIAAACkeFAAAAAAAAAAAAAAAAHQAAAAAAAAAAAAAAACaGQAAQ29udmVydGVyL1dlYkNvbnRl" >> mini-ide.jar.b64
echo "bnQvV0VCLUlORi9QSwECCgAKAAAIAABcqMZAAAAAAAAAAAAAAAAJQAAAAAAAAAAAAAAADVGQAA" >> mini-ide.jar.b64
echo "Q29udmVydGVyL1dlYkNvbnRlbnQvV0VCLUlORi9jbGFzc2VzL1BLAQIUABQACAgIAGKN4UBDhyA7" >> mini-ide.jar.b64
echo "AAEAAGQCAAAkAAAAAAAAAAAAAAAABgaAABDb252ZXJ0ZXIvV2ViQ29udGVudC9XRUItSU5GL3dl" >> mini-ide.jar.b64
echo "Yi54bWxQSwECFAAUAAgICABWjeFADF+nL+4AAAAfAgAAKQAAAAAAAAAAAAAAABqGwAAQ29udmVy" >> mini-ide.jar.b64
echo "dGVyL1dlYkNvbnRlbnQvV0VCLUlORi93ZWJsb2dpYy54bWxQSwUGAAAAABEAEQATBQAArxwAAAAA" >> mini-ide.jar.b64
```

```
cat > README.TXT << __EOF__
Use the following commands to unpack and install.
On LINUX (tested on Linux  2.6.18-194.el5 #1 SMP Mon Mar 29 20:06:41 EDT 2010 i686 i686 i386 GNU/Linux)
==================================================
base64 -i -d mini-ide.jar.b64 > mini-ide.jar
jar xvf mini-ide.jar
rm -f mini-ide.jar.b64
find . -name '*.sh' -exec dos2unix {} \;

On MAC OSX
```

```
================================================
base64 -i mini-ide.jar.b64 -D > mini-ide.jar
rm  mini-ide.jar.b64
jar xvf mini-ide.jar

__EOF__


cat README.TXT
```