Rhino J2EE Examples User Guide

December 15, 2010

Open Cloud Limited
Level 4, 54-56 Cambridge Terrace
Wellington
New Zealand
http://www.opencloud.com

Contents

1 Introduction

The following document explains how to setup, configure and use the Rhino-SLEE<->J2EE Integration example applications that are distributed with the Rhino releases. After running and completing the Rhino SDK or Production install script the J2EE examples' base directory can be found in $RHINO_HOME/examples/j2ee (with $RHINO_HOME standing for the path under which the Rhino distribution has been installed). This directory will further be referred to as $J2EE_HOME.

In $J2EE_HOME/lib you should find the following archives and files:

1. test-j2ee-callejb.ear - J2EE Application for deployment in JBoss 4.2.2, containing the SLEEConnector (rhino-j2ee-connector.rar), AccountEJB (test-j2ee-account-ejb.jar) and Account Web Application (test-j2ee-callejb-war.war)
2. test-j2ee-callejb-weblogic.ear - J2EE Application for deployment in WebLogic 10.0, containing the SLEEConnector (rhino-j2ee-connector.rar), AccountEJB (test-j2ee-account-ejb.jar) and Account Web Application (test-j2ee-callejb-war.war)
3. jboss-ds.xml - JBoss specific Deployment Descriptor for the SLEEConnector .rar module.

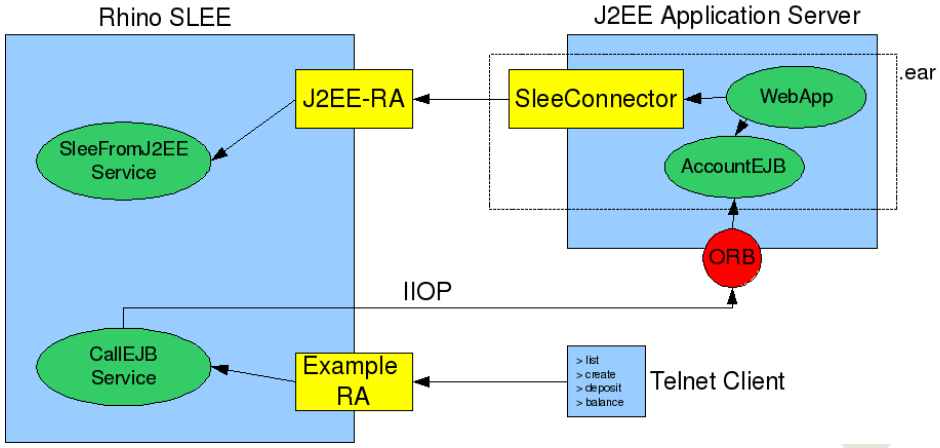
In $J2EE_HOME/du you should find the following archives and files:

1. example-ra-du_2.2.0.jar - Deployable Unit containing the OpenCloud Example RA for deployment into Rhino
2. rhino-j2ee-connector-ra-du_2.1.0.jar - Deployable Unit containing the OpenCloud J2EE Connector RA for deployment into Rhino
3. test-j2ee-callejb-service-du.jar - Deployable Unit containing the 'Open Cloud Sbb EJB integration test' service (dependent on Example RA) for deployment into Rhino
4. test-sleefromj2ee-service-du.jar - Deployable Unit containing containing the 'Open Cloud J2EE to SLEE integration test' service and events for deployment into Rhino

The J2EE examples currently contain distributions for JBoss Application Server 4.2.2 and BEA WebLogic 10.0 and have been tested with these platforms on Linux, Solaris and Windows XP.

Also note, that the examples are currently configured assuming that Rhino SLEE and J2EE Application Server are running on the same host (i.e. all connection configurations are referring to 'localhost').

The following diagram shows the general architecture of the J2EE examples application:



2 Running J2EE examples with JBoss application server

2.1 Starting the JBoss application server

Download JBoss application server (version 4.2.2, `jboss-4.2.2.tar.bz2`) from `www.jboss.org`, extract the package and copy the contents to an installation directory of your choice (e.g. `/home/userXY/jboss` or `c:/jboss`) which will further be referred to as `$JBOSS_HOME`.

The Rhino J2EE examples use IIOP for accessing J2EE server components from SLEE Sbbs, thus JBoss has to be started in the "-all" configuration (in "-default" the IIOP modules are disabled).

```
$ JBOSS_HOME/bin>./run.sh -c all
```

To shutdown JBoss run the `shutdown.sh` script (or press `Ctrl+C` which terminates the VM).

After JBoss has started up, you can access the administration console under `http://localhost:8080`

2.2 Configuring Rhino for use with JBoss

Rhino uses IIOP to access EJB components on the J2EE Application Server side. For this to work appropriately you have to modify the `rhino-config.xml` file in `$RHINO_HOME/config` (or `$RHINO_HOME/<node-directory>/config` for the Rhino Production version).

Add the `<ejb-resources>` tag shown below, with `corbaname::localhost:3528/JBoss/Naming/root#Account` as `<remote-url>`, the result should look like:

```
...
    </jdbc>
  </ocbb-resources>
  <ejb-resources>
    <remote-ejb>
      <ejb-name>ejb:AccountHome</ejb-name>
      <home>com.opencloud.slee.example.j2ee.callejb.ejb.AccountHome</home>
      <remote-url>corbaname::localhost:3528/JBoss/Naming/root#Account</remote-url>
    </remote-ejb>
  </ejb-resources>
</rhino-config>
```

Now startup the Rhino SLEE with `./start-rhino.sh`.

2.3 Deploying the J2EE Application components

Copy `test-j2ee-callejb.ear` from `$J2EE_HOME/lib` to the JBoss deployment directory `$JBOSS_HOME/server/all/deploy`. The following messages will appear in the JBoss console window.

```
15:49:22,634 INFO  [EARDeployer] Init J2EE application: file:/home/tspall/jboss-4.2.2.GA/server/
                   all/deploy/test-j2ee-callejb.ear
15:49:22,770 INFO  [EjbModule] Deploying Account
15:49:22,923 INFO  [RARDeployment] Required license terms exist, view META-INF/ra.xml in .../tmp/
                   deploy/tmp25295test-j2ee-callejb.ear-contents/rhino-j2ee-connector.rar
15:49:23,019 INFO  [Account] EJBHome reference for Account:
                   IOR:000000000000004D52...
15:49:23,019 INFO  [Account] Home IOR for Account bound to iiop/Account in JNP naming service
15:49:23,023 INFO  [Account] Home IOR for Account bound to Account in CORBA naming service
15:49:23,071 INFO  [ProxyFactory] Bound EJB Home 'Account' to jndi 'Account'
15:49:23,072 INFO  [EJBDeployer] Deployed: file:/home/tspall/jboss-4.2.2.GA/server/all/tmp/deploy/
                   tmp25295test-j2ee-callejb.ear-contents/test-j2ee-account-ejb.jar
15:49:23,081 INFO  [TomcatDeployer] deploy, ctxPath=/slee, warUrl=.../tmp/deploy/
                   tmp25295test-j2ee-callejb.ear-contents/test-j2ee-callejb-war-exp.war/
15:49:23,147 INFO  [EARDeployer] Started J2EE application: file:/home/tspall/jboss-4.2.2.GA/server/
                   all/deploy/test-j2ee-callejb.ear
```

Then copy `jboss-ds.xml` from `$J2EE_HOME/lib` into the deployment directory.

```
<!DOCTYPE connection-factories PUBLIC
          "-//JBoss//DTD JBOSS JCA Config 1.5//EN"
          "http://www.jboss.org/j2ee/dtd/jboss-ds_1_5.dtd">

<connection-factories>
   <no-tx-connection-factory>
      <jndi-name>RhinoConnection</jndi-name>
      <rar-name>test-j2ee-callejb.ear#rhino-j2ee-connector.rar</rar-name>
      <connection-definition>javax.slee.connection.SleeConnectionFactory</connection-definition>
      <config-property name="endpoints" type="java.lang.String">localhost:12345</config-property>
   </no-tx-connection-factory>
</connection-factories>
```

The following messages will appear on the JBoss console window to confirm the creation and registration of the SleeConnectionFactory.

```
15:51:13,382 INFO  [RhinoConnection] com.opencloud.slee.resources.connector.impl.ManagedConnectionFactory
                   Impl@46a171 : createConnectionFactory(org.jboss.resource.connectionmanager.
                   BaseConnectionManager2$ConnectionManagerProxy@f20964)
15:51:13,387 INFO  [RhinoConnection] com.opencloud.slee.resources.connector.impl.ManagedConnectionFactory
                   Impl@46a171 : createConnectionFactory(org.jboss.resource.connectionmanager.
                   BaseConnectionManager2$ConnectionManagerProxy@f20964) = com.opencloud.slee.
                   resources.connector.impl.SleeConnectionFactoryImpl@fc63be
15:51:13,390 INFO  [ConnectionFactoryBindingService] Bound ConnectionManager
                   'jboss.jca:service=ConnectionFactoryBinding,name=RhinoConnection'
                   to JNDI name 'java:RhinoConnection'
```

2.4 Installing the J2EE Connector RA into Rhino

Connect the Rhino Command Console (`$RHINO_HOME/client/bin/rhino-console`). Run the following commands to install the J2EE Connector RA and create and activate a new RA entity "j2eera".

```
[Rhino@localhost (#0)] installlocaldu $J2EE_HOME/lib/rhino-j2ee-connector-ratype-du_2.1.0.jar
...
[Rhino@localhost (#1)] installlocaldu $J2EE_HOME/lib/rhino-j2ee-connector-ra-du_2.1.0.jar
...
[Rhino@localhost (#2)] createraentity name=J2EE\ Connector\ Receiver,vendor=OpenCloud,
                       version=2.1 j2eera
...
[Rhino@localhost (#3)] activateraentity j2eera
...
```

The RA entity's "port" property is initialized with "12345" and thus automatically matches the configuration for the SLEEConnector component on J2EE side.

2.5 Installing Example RA and the J2EE Integration test services

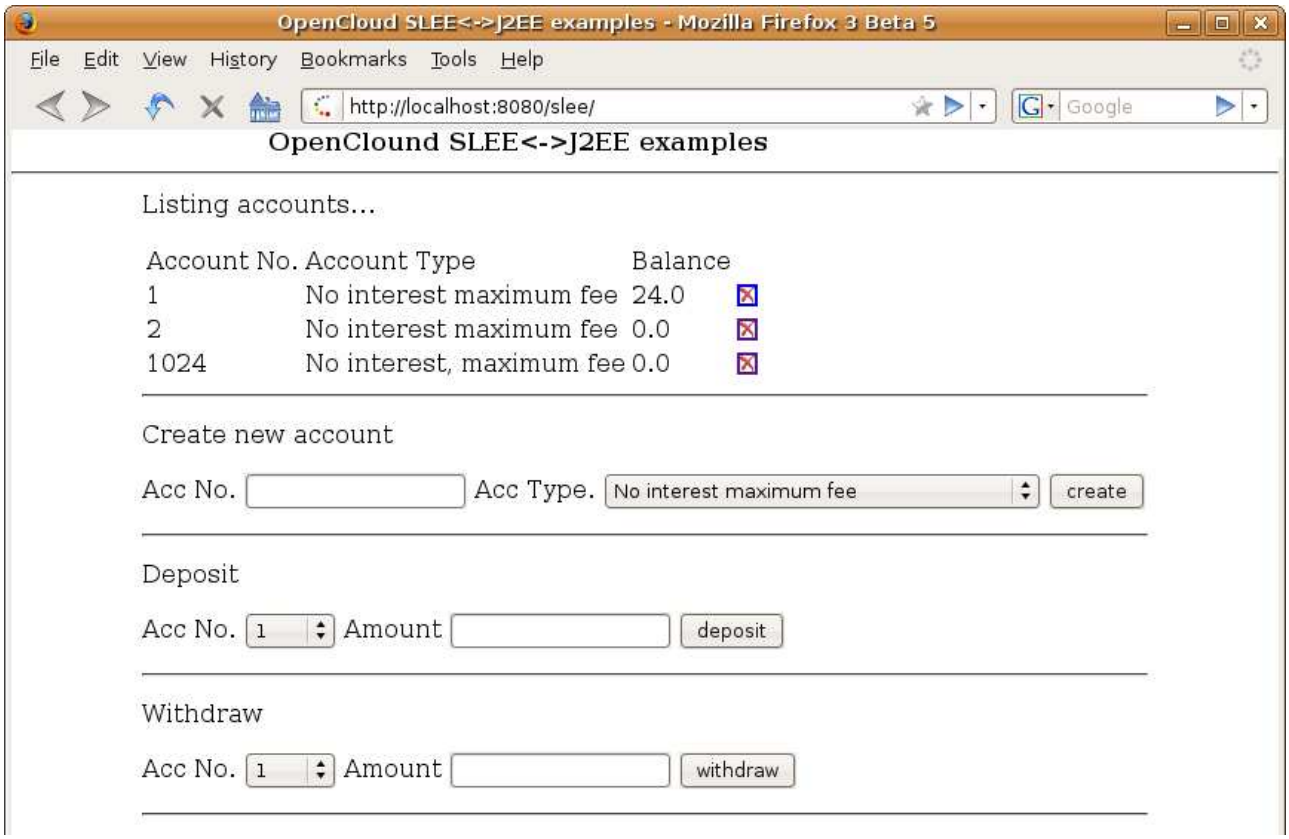
On the Rhino Command Console run the following commands to install the J2EE examples' remaining components.

```
[Rhino@localhost (#4)] installlocaldu $J2EE_HOME/lib/example-ra-du_2.2.0.jar
...
[Rhino@localhost (#5)] createraentity name=Example,vendor=OpenCloud,version=2.0 examplera
...
[Rhino@localhost (#6)] activateraentity examplera
...
[Rhino@localhost (#7)] installlocaldu $J2EE_HOME/lib/test-j2ee-callejb-service-du.jar
...
[Rhino@localhost (#8)] activateservice name=Open\ Cloud\ Sbb\ EJB\ integration\ test,
                       vendor=OpenCloud,version=2.1
...
[Rhino@localhost (#10)] installlocaldu $J2EE_HOME/lib/test-sleefromj2ee-service-du.jar
...
[Rhino@localhost (#11)] activateservice name=Open\ Cloud\ J2EE\ to\ SLEE\ integration\ test,
                        vendor=OpenCloud,version=2.1
...
```

2.6 Connection to the test JBoss web application

Accessing `http://localhost:8080/slee` with a browser (e.g. Firefox, Internet Explorer) now lists a survey of the Account entities existing in the J2EE Application Server. The accounts can be manipulated (created, removed, amounts

deposited, withdrawn) via the displayed buttons and text fields. Each time a modification is done an appropriate event is sent to the Rhino SLEE via the SLEEConnector.



The "Open Cloud J2EE to SLEE integration test" service SBBs receive these events and react by printing out trace messages like the following on the Rhino log output:

```
2008-04-21 17:11:27.461  INFO   [trace.]   <J2EE-RA Listener/12345> [entity=j2eera]  New
                         connection from /127.0.0.1:53344
2008-04-21 17:11:27.519  INFO   [trace.]   <jr-0> [service=ServiceID[name=Open Cloud J2EE
                         to SLEE integration test,
vendor=OpenCloud,version=1.1],sbb=SbbID[name=fromj2ee,vendor=OpenCloud,version=1.1]]
Received IntegrationEvent on activity NullActivity[ExternalActivityHandle(/192.168.0.61,
23fe778a:1196f43ae70:-7fff)]:
    type:    3
    message: Deposited amount=2.0 into account accountNo=3
```

2.7 Manipulating accounts from SLEE side

Use a telnet client and connect to the Example RA.

```
$ telnet localhost 9999
```

The "Open Cloud Sbb EJB integration test" service supports the following commands as can also be seen by typing "help" in the telnet window:

```
tspall@submarine:~$ telnet localhost 9999
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
help

************ Usage for callejb test: ************
Commands:
    list                        : Retrieves survey of all accounts
    create <accNo>              : Creates new account with account number 'accNo'(int)
    remove <accNo>              : Removes the account with account number 'accNo' if it exists
    deposit <accNo> <amount>    : Deposits 'amount'(double) into account 'accNo'(int)
    withdraw <accNo> <amount>   : Withdraws 'amount'(double) from account 'accNo'(int)
    balance <accNo>             : Displays the current balance of account 'accNo'(int)
    help                        : Displays this survey
    close                       : To close the telnet connection and exit.

************************************************
```

Try e.g. creating a new account "1234" and making a deposit by typing `create 1234`, then `deposit 1234 5.0`.

```
list
*********** Account survey list ***************
    Acc no.           Acc type                                  Balance
    1 No interest maximum fee                          7.23
 2048 MyType                                           0.0

create 1234
Created account 1234
deposit 1234 5.0
Deposited 5.0 into account 1234
list
*********** Account survey list ***************
    Acc no.           Acc type                                  Balance
    1 No interest maximum fee                          7.23
 1234 No interest, maximum fee                         5.0
 2048 MyType                                           0.0
```

3 Running J2EE exapmples with WebLogic application server

3.1 Installing BEA WebLogic Application Server

These examples have been written to work with WebLogic 10.0. You can download BEA WebLogic Application Server (version 10.01) from `www.bea.com`. If you are running Ubuntu Linux (which is not officially supported with WebLogic) just try the RedHat version.

Follow the installation instructions and install WebLogic Application Server in `/home/<username>/bea` (or similar). This directory will further be referred to as `$WL_HOME`.

3.2 Configuring BEA WebLogic Application Server

Before starting the WebLogic server, we need to create a new working domain. Open a terminal, go to `$WL_HOME/wlserver_10.0/common/bin` directory and run the `config.sh` script.

```
$ WL_HOME/wlserver_10.0/common/bin> ./config.sh
```

Follow the instructions on the screen to create a new domain "mydomain" based on the standard template. For detailed information, please refer to Appendix A.

To be able to deploy the J2EE examples in WebLogic you need a Database server installation and appropriate JDBC drivers. (BEA WebLogic does not come with an integrated default DB like e.g. JBoss) For these examples we will use PostgreSQL as our database of choice. However, you can use any of the databases supported by WebLogic (e.g. MySQL, Sybase etc.) and set them up accordingly.

Please refer to Appendix B of this document to learn about where to get PostgreSQL DB and how to install the appropriate JDBC drivers for WebLogic.

Before starting the deployment you must also prepopulate the DB installation you created with the database and the schema and table used by the examples. Appendix D outlines how to do this in case your DB installation is PostgreSQL.

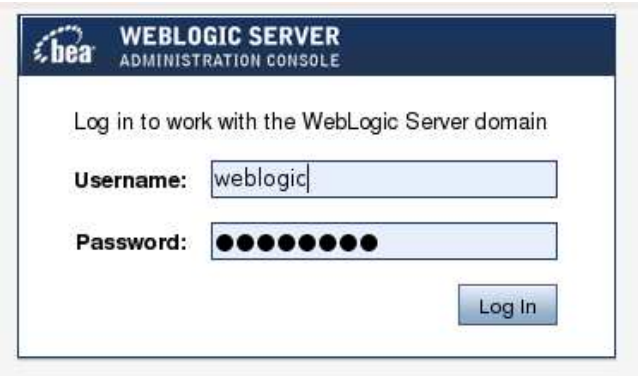
3.3 Starting BEA WebLogic Application Server

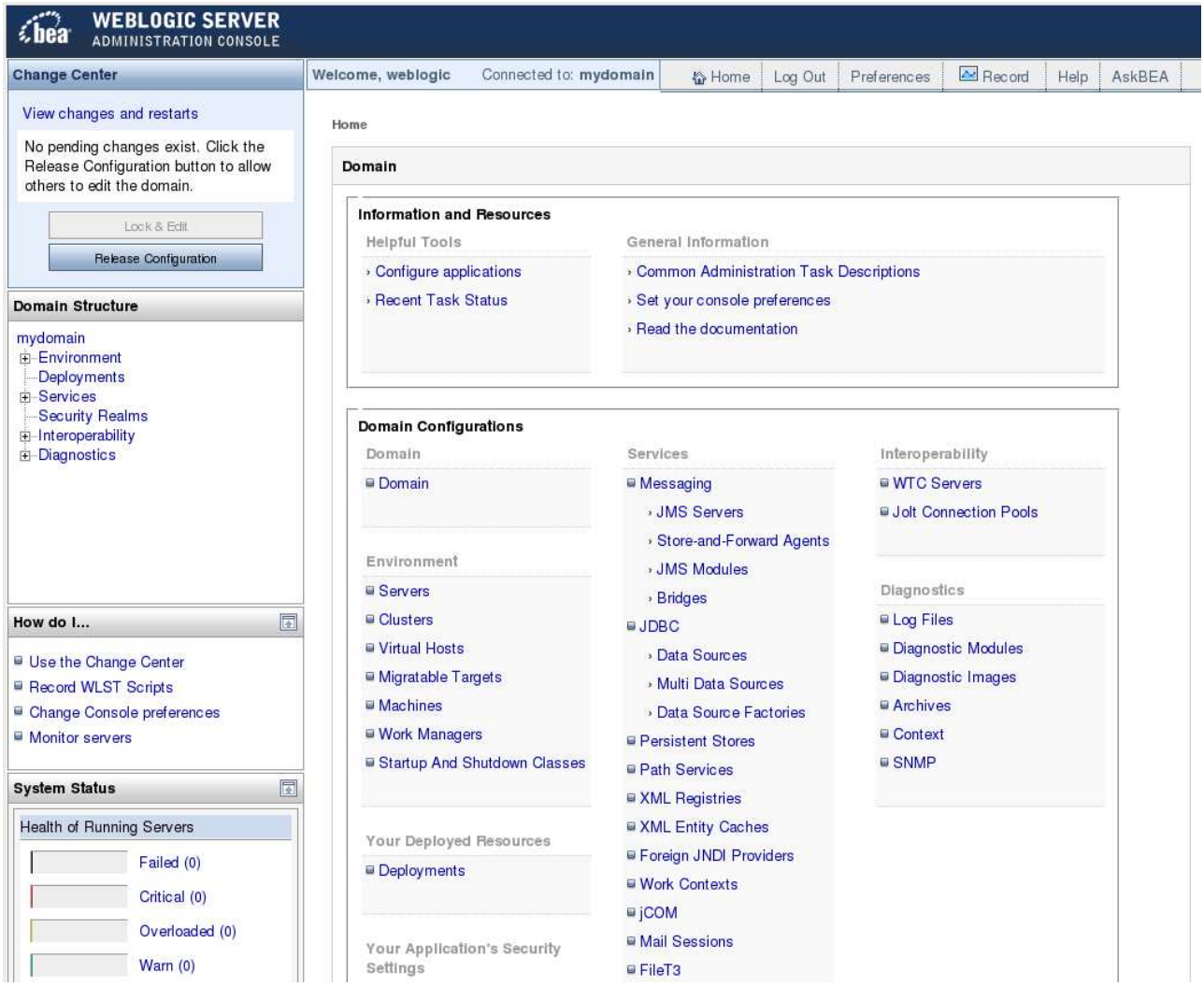
Having a Database server available, JDBC drivers for WebLogic installed and the DB prepopulated with the necessary data, now startup the Application Server.

```
$ WL_HOME/user_projects/domains/mydomain>./startWebLogic.sh
```

To shutdown WebLogic run the `stopWebLogic.sh` script (or press `Ctrl+C` which terminates the VM).

After WebLogic has started up, you can access the administration console under `http://localhost:7001/console`.





3.4 Configuring Rhino for use with BEA WebLogic

Rhino uses IIOP to access EJB components on the J2EE Application Server side. For this to work appropriately you have to modify the `rhino-config.xml` file in `$RHINO_HOME/config` (or `$RHINO_HOME/<node-directory>/config` for the Rhino Production version).

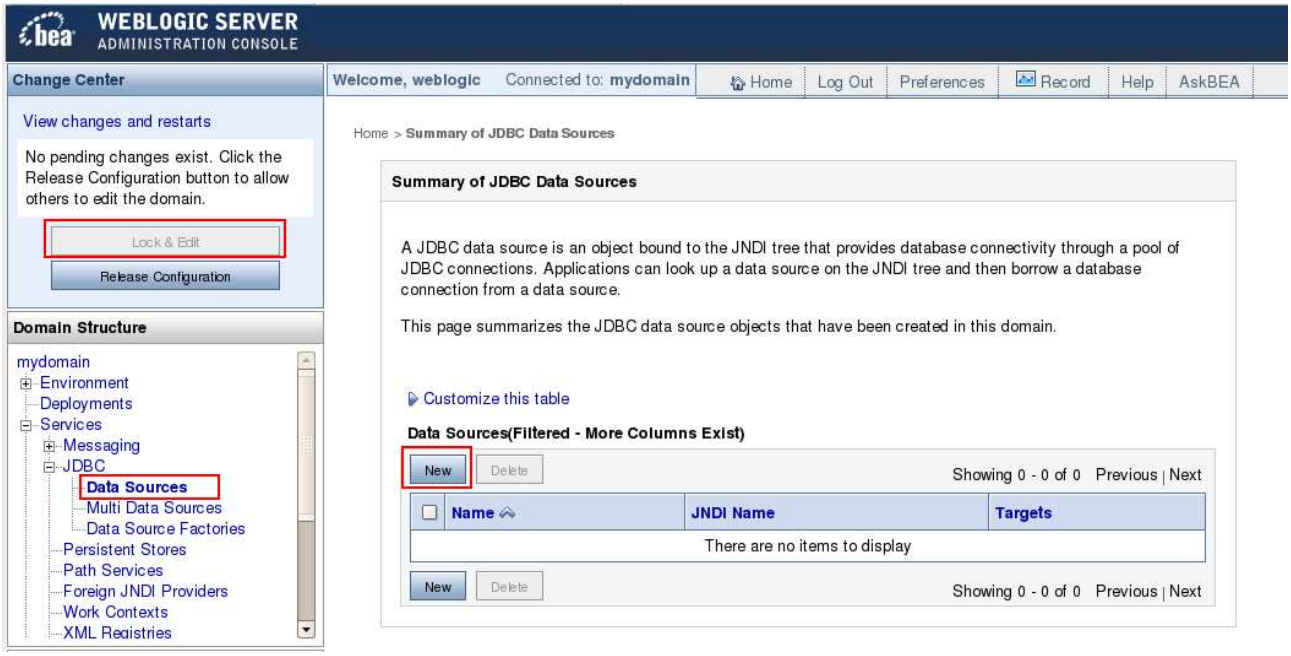
Add the `<ejb-resources>` tag shown below, with `iiop://localhost:7001/Account` as `<remote-url>`, the result should look like:

```
...
    </jdbc>
  </ocbb-resources>
  <ejb-resources>
    <remote-ejb>
      <ejb-name>ejb:AccountHome</ejb-name>
      <home>com.opencloud.slee.example.j2ee.callejb.ejb.AccountHome</home>
      <remote-url>iiop://localhost:7001/Account</remote-url>
    </remote-ejb>
  </ejb-resources>
</rhino-config>
```

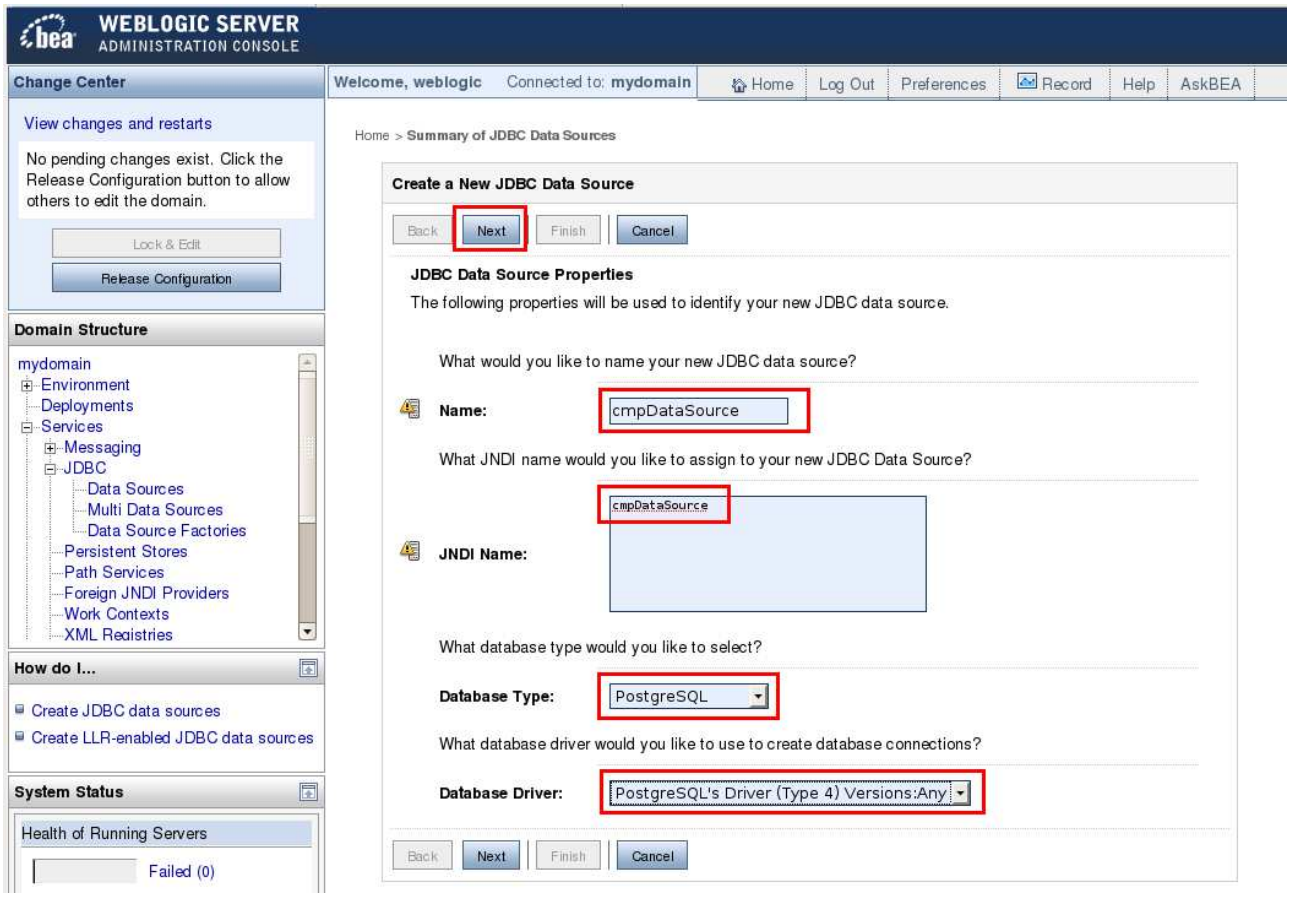
Now startup the Rhino SLEE with `./start-rhino.sh`.

3.5 Setup WebLogic JDBC pool

In the WebLogic console menu, open up "`myDomain -> Services -> JDBC -> Data Sources`", click on "`Lock & Edit`" and create a new JDBC Data Source that connects to your Database installation.



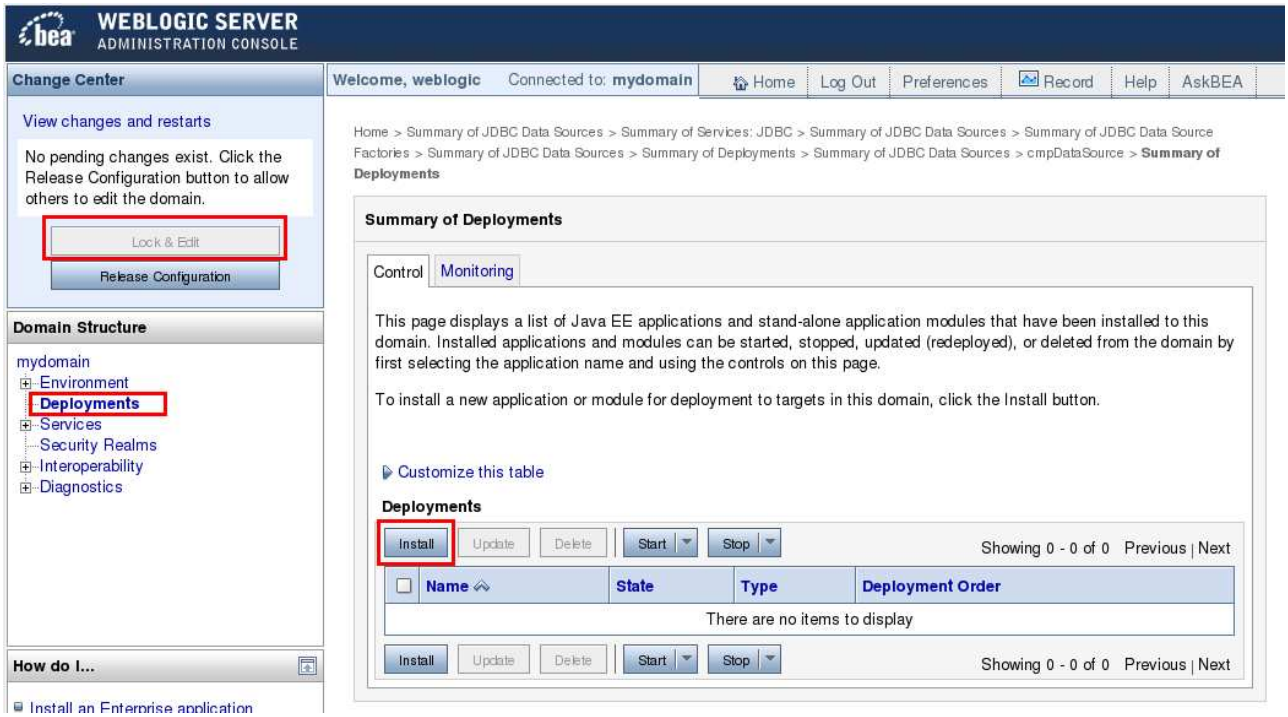
As JNDI name for the new data source you have to enter "cmpDataSource".



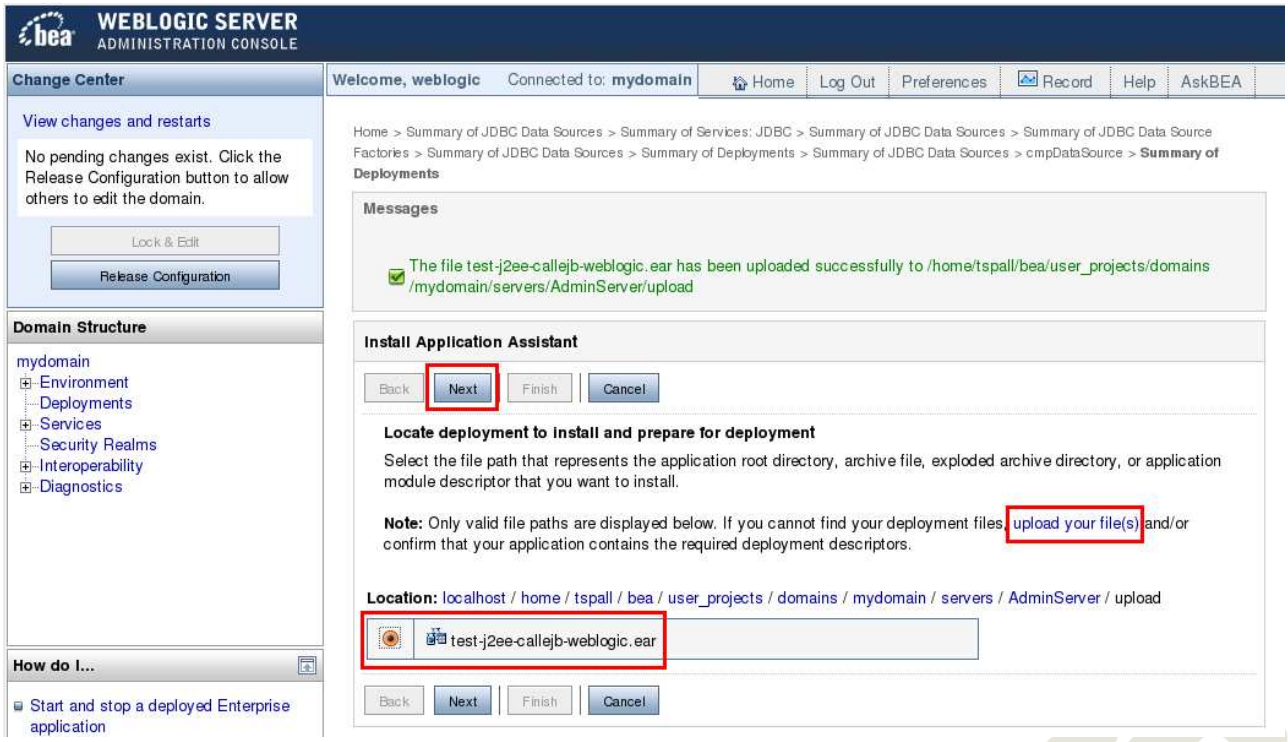
Please refer to Appendix C for detailed instructions on how to go through the configuration wizard for our example PostgreSQL DB installation.

3.6 Deploying and starting the J2EE Application components

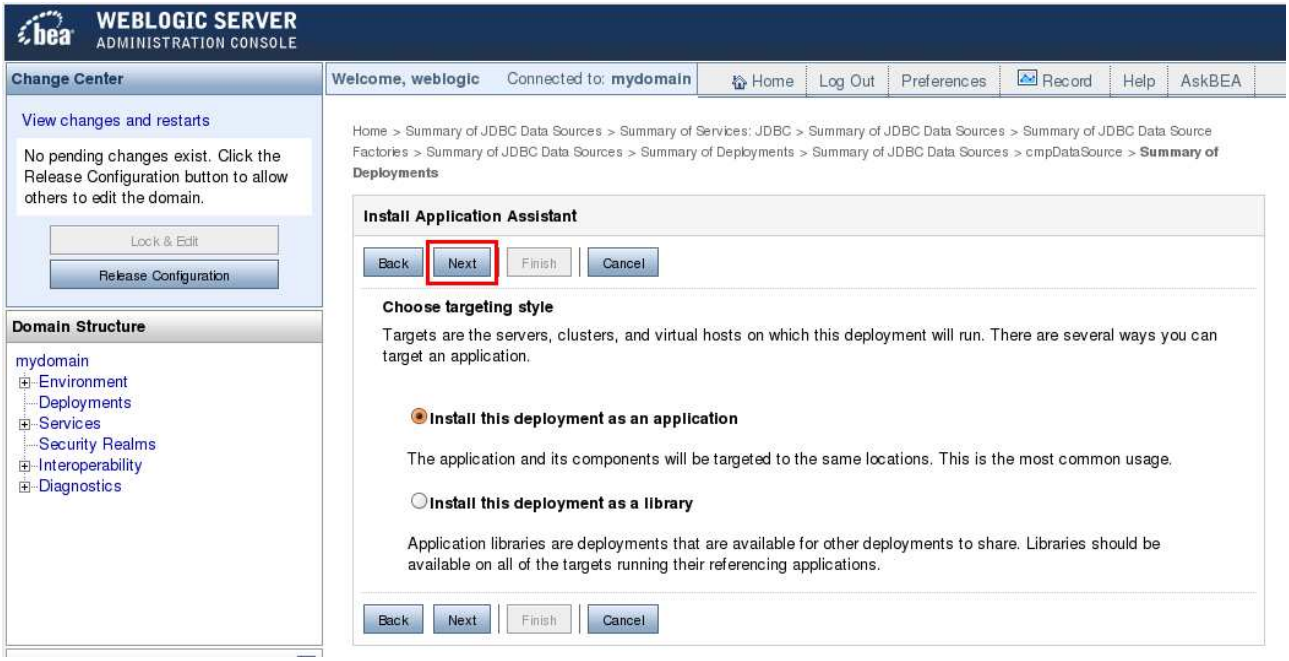
In the WebLogic console menu, open up "`myDomain -> Deployments`", click on "`Lock & Edit`" and then on "`Install`".

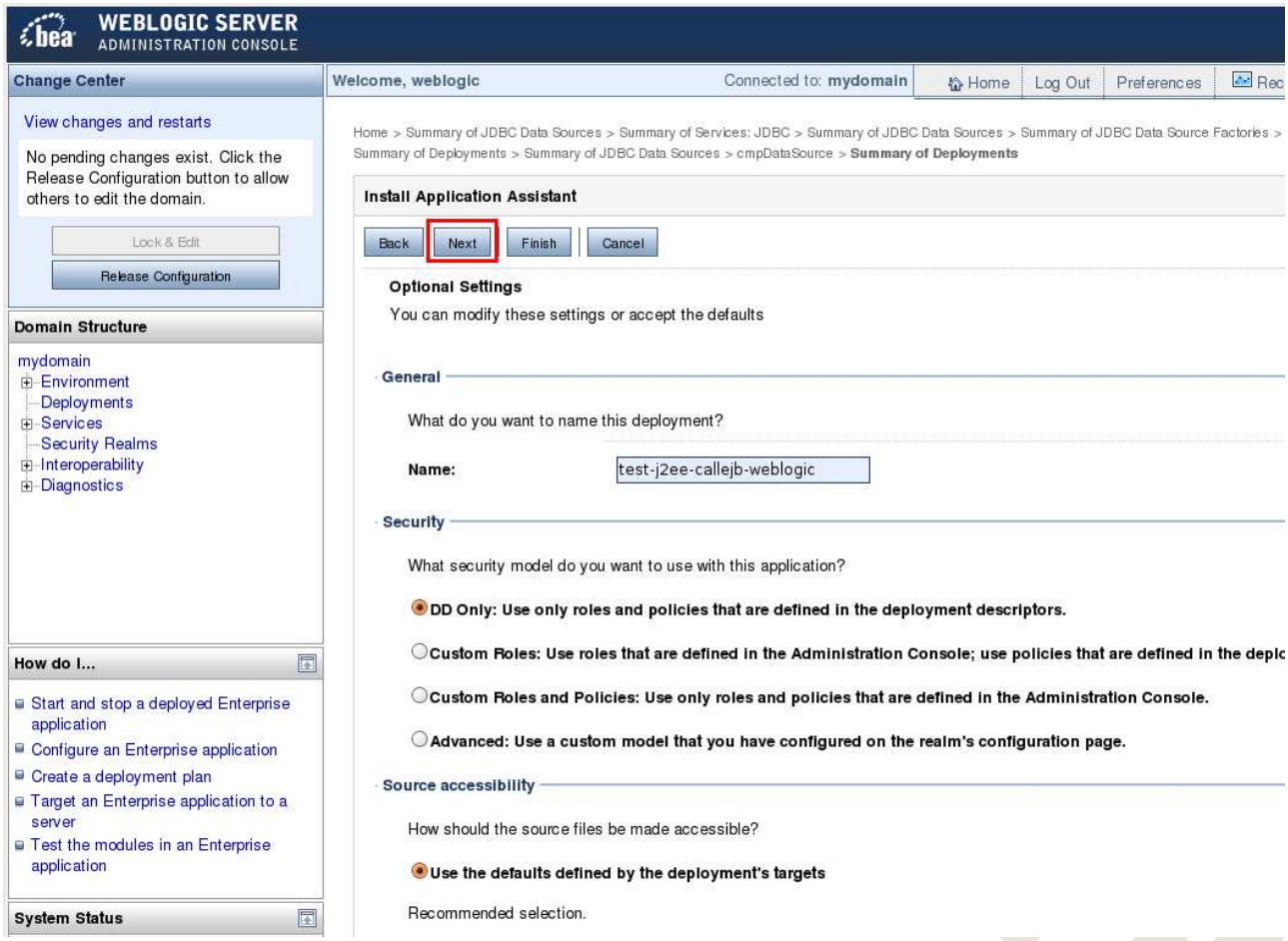


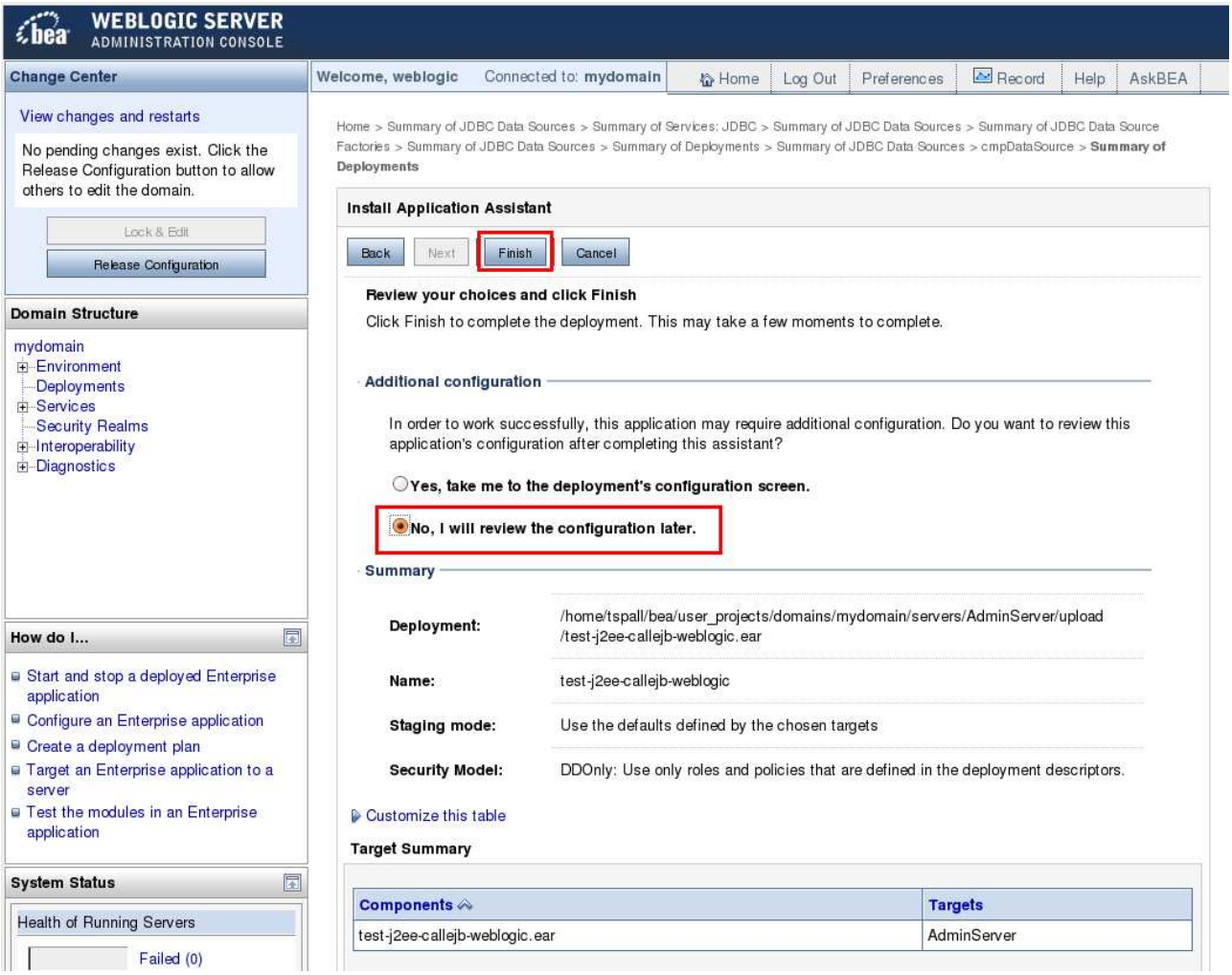
Upload the `test-j2ee-callejb-weblogic.ear` file from the `$J2EE_HOME/lib` directory, then click on `"Next"`.



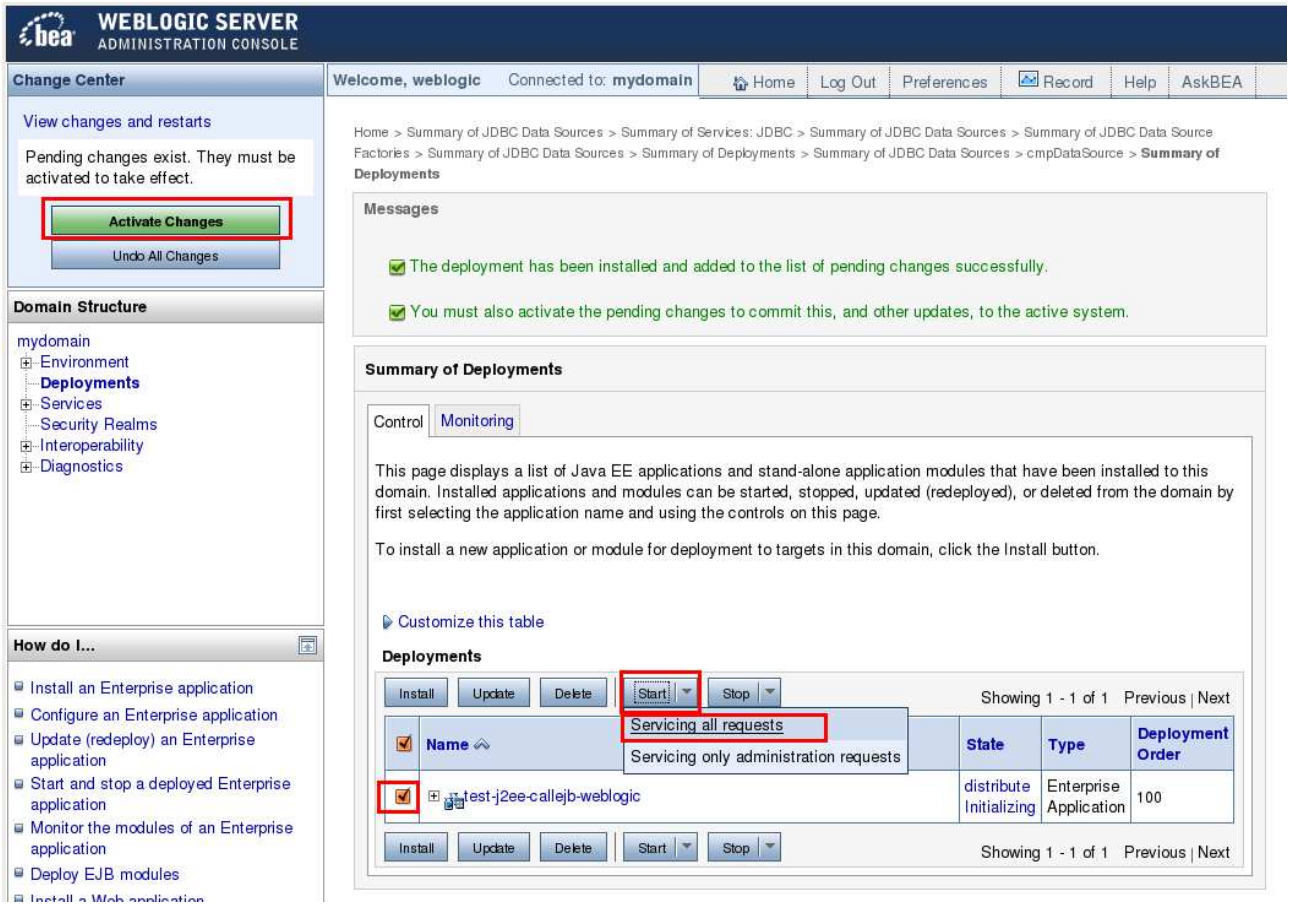
In the following wizard windows you can mostly just leave the default values as they are and click on `"Next"`. Only change the review setting to "No, I will review the configuration later".


WEBLOGIC SERVER
ADMINISTRATION CONSOLE
Change Center
View changes and restarts
No pending changes exist. Click the Release Configuration button to allow others to edit the domain.
Lock & Edit
Release Configuration
Domain Structure
mydomain
Environment
Deployments
Services
Security Realms
Interoperability
Diagnostics
Welcome, weblogic
Connected to: mydomain
Home
Log Out
Preferences
Record
Help
AskBEA
Home > Summary of JDBC Data Sources > Summary of Services: JDBC > Summary of JDBC Data Sources > Summary of JDBC Data Source Factories > Summary of JDBC Data Sources > Summary of Deployments > Summary of JDBC Data Sources > cmpDataSource > Summary of Deployments
Install Application Assistant
Back
Next
Finish
Cancel
Choose targeting style
Targets are the servers, clusters, and virtual hosts on which this deployment will run. There are several ways you can target an application.
Install this deployment as an application
The application and its components will be targeted to the same locations. This is the most common usage.
Install this deployment as a library
Application libraries are deployments that are available for other deployments to share. Libraries should be available on all of the targets running their referencing applications.
Back
Next
Finish
Cancel


WEBLOGIC SERVER
ADMINISTRATION CONSOLE
Change Center
View changes and restarts
No pending changes exist. Click the Release Configuration button to allow others to edit the domain.
Lock & Edit
Release Configuration
Domain Structure
mydomain
Environment
Deployments
Services
Security Realms
Interoperability
Diagnostics
How do I...
Start and stop a deployed Enterprise application
Configure an Enterprise application
Create a deployment plan
Target an Enterprise application to a server
Test the modules in an Enterprise application
System Status
Welcome, weblogic
Connected to: mydomain
Home
Log Out
Preferences
Home > Summary of JDBC Data Sources > Summary of Services: JDBC > Summary of JDBC Data Sources > Summary of JDBC Data Source Factories > Summary of Deployments > Summary of JDBC Data Sources > cmpDataSource > Summary of Deployments
Install Application Assistant
Back
Next
Finish
Cancel
Optional Settings
You can modify these settings or accept the defaults
General
What do you want to name this deployment?
Name:
test-j2ee-callejb-weblogic
Security
What security model do you want to use with this application?
DD Only: Use only roles and policies that are defined in the deployment descriptors.
Custom Roles: Use roles that are defined in the Administration Console; use policies that are defined in the deplo
Custom Roles and Policies: Use only roles and policies that are defined in the Administration Console.
Advanced: Use a custom model that you have configured on the realm's configuration page.
Source accessibility
How should the source files be made accessible?
Use the defaults defined by the deployment's targets
Recommended selection.



After the deployment has been completed successfully, you start the application and finally click on "`Activate Changes`".



3.7 Installing the J2EE Connector RA into Rhino

Connect the Rhino Command Console (`$RHINO_HOME/client/bin/rhino-console`). Run the following commands to install the J2EE Connector RA and create and activate a new RA entity "j2eera".

```
[Rhino@localhost (#0)] installlocaldu $J2EE_HOME/lib/rhino-j2ee-connector-ratype-du_2.1.0.jar
...
[Rhino@localhost (#1)] installlocaldu $J2EE_HOME/lib/rhino-j2ee-connector-ra-du_2.1.0.jar
...
[Rhino@localhost (#2)] createraentity name=J2EE\ Connector\ Receiver,vendor=OpenCloud,
                       version=2.1 j2eera
...
[Rhino@localhost (#3)] activateraentity j2eera
...
```

The RA entity's "port" property is initialized with "12345" and thus automatically matches the configuration for the SLEEConnector component on J2EE side.

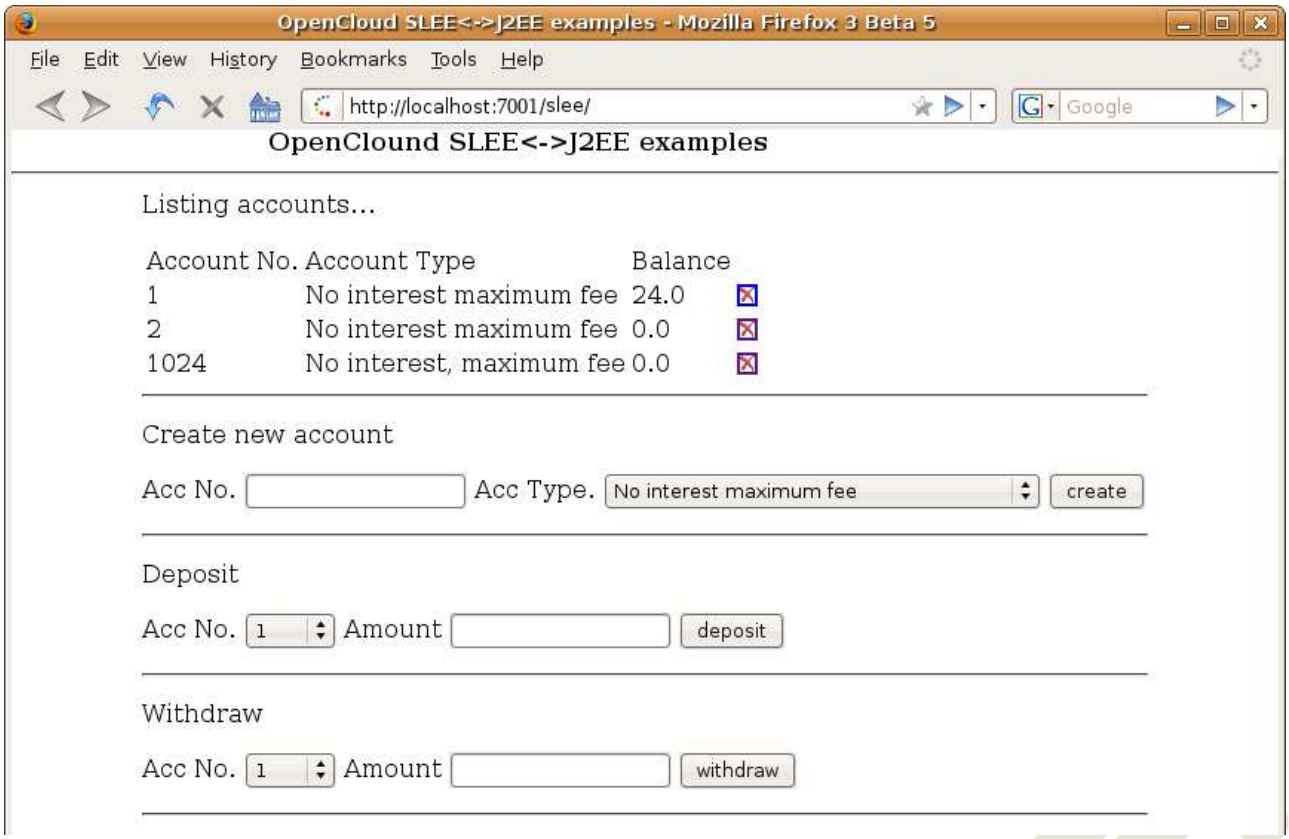
3.8 Installing Example RA and the J2EE Integration test services

On the Rhino Command Console run the following commands to install the J2EE examples' remaining components.

```
[Rhino@localhost (#4)] installlocaldu $J2EE_HOME/lib/example-ra-du_2.2.0.jar
...
[Rhino@localhost (#5)] createraentity name=Example,vendor=OpenCloud,version=2.0 examplera
...
[Rhino@localhost (#6)] activateraentity examplera
...
[Rhino@localhost (#7)] installlocaldu $J2EE_HOME/lib/test-j2ee-callejb-service-du.jar
...
[Rhino@localhost (#8)] activateservice name=Open\ Cloud\ Sbb\ EJB\ integration\ test,
                       vendor=OpenCloud,version=2.1
...
[Rhino@localhost (#10)] installlocaldu $J2EE_HOME/lib/test-sleefromj2ee-service-du.jar
...
[Rhino@localhost (#11)] activateservice name=Open\ Cloud\ J2EE\ to\ SLEE\ integration\ test,
                        vendor=OpenCloud,version=2.1
...
```

3.9 Connection to the test WebLogic web application

Accessing `http://localhost:7001/slee` now lists a survey of the Account entities existing in the J2EE Application Server. The accounts can be manipulated (created, removed, amounts deposited, withdrawn) via the displayed buttons and text fields. Each time a modification is done an appropriate event is sent to the Rhino SLEE via the SLEEConnector.



The "Open Cloud J2EE to SLEE integration test" service sbbs receive these events and react by printing out trace messages like the following on the Rhino log output:

```
2008-04-21 17:11:27.461  INFO   [trace.]   <J2EE-RA Listener/12345> [entity=j2eera]  New
                         connection from /127.0.0.1:53344
2008-04-21 17:11:27.519  INFO   [trace.]   <jr-0> [service=ServiceID[name=Open Cloud J2EE
                         to SLEE integration test,
vendor=OpenCloud,version=1.1],sbb=SbbID[name=fromj2ee,vendor=OpenCloud,version=1.1]]
Received IntegrationEvent on activity NullActivity[ExternalActivityHandle(/192.168.0.61,
23fe778a:1196f43ae70:-7fff)]:
    type:    3
    message: Deposited amount=2.0 into account accountNo=3
```

3.10 Manipulating accounts from SLEE side

Use a telnet client and connect to the Example RA.

```
$ telnet localhost 9999
```

The "Open Cloud Sbb EJB integration test" service supports the following commands as can also be seen by typing `help` in the telnet window:

```
tspall@submarine:~$ telnet localhost 9999
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
help

************ Usage for callejb test: ************
Commands:
    list                        : Retrieves survey of all accounts
    create <accNo>              : Creates new account with account number 'accNo'(int)
    remove <accNo>              : Removes the account with account number 'accNo' if it exists
    deposit <accNo> <amount>    : Deposits 'amount'(double) into account 'accNo'(int)
    withdraw <accNo> <amount>   : Withdraws 'amount'(double) from account 'accNo'(int)
    balance <accNo>             : Displays the current balance of account 'accNo'(int)
    help                        : Displays this survey
    close                       : To close the telnet connection and exit.

************************************************
```

Try e.g. creating a new account "1234" and making a deposit by typing `create 1234`, then `deposit 1234 5.0`.

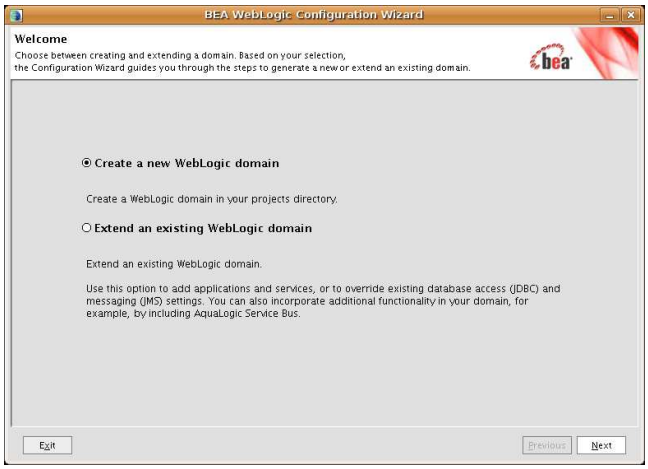
```
list
************ Account survey list ***************
    Acc no.           Acc type                                  Balance
    1 No interest maximum fee                          7.23
 2048 MyType                                           0.0

create 1234
Created account 1234
deposit 1234 5.0
Deposited 5.0 into account 1234
list
************ Account survey list ***************
    Acc no.           Acc type                                  Balance
    1 No interest maximum fee                          7.23
 1234 No interest, maximum fee                         5.0
 2048 MyType                                           0.0
```

A Creating a new WebLogic domain from the standard template

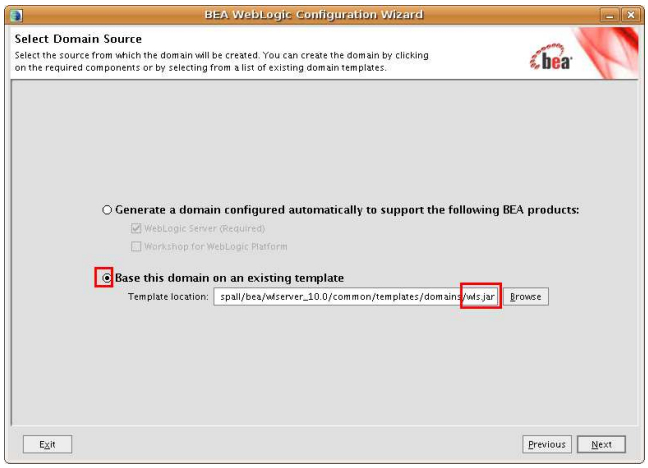
This appendix describes in detail the steps necessary to create a new user domain in WebLogic Application Server 10.0. After installing WebLogic Application Server 10, go to its home directory $WL_HOME and run the following script:

```
$ > $WL_HOME/wlserver_10.0/common/bin/config.sh
```

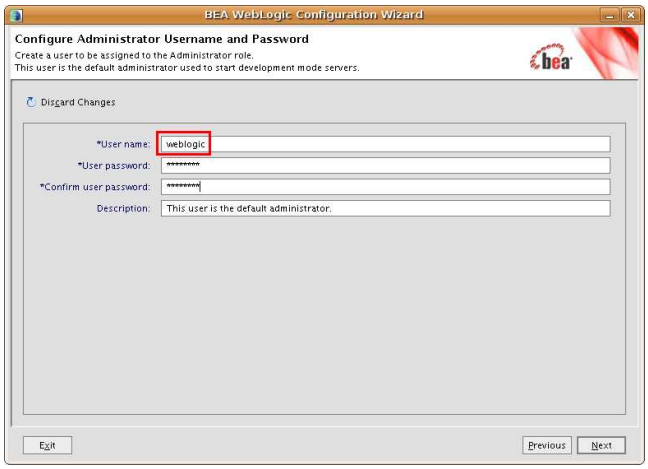
A configuration wizard window opens up.



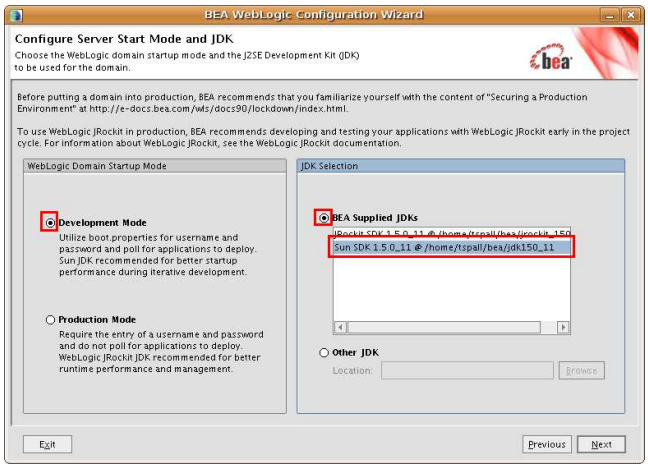
Base the new domain on an existing template and choose "wls.jar" as the template file.



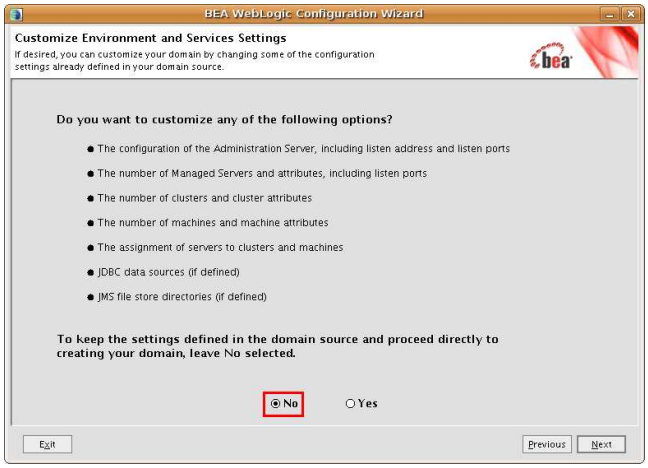
Choose "weblogic" as username and enter a password.



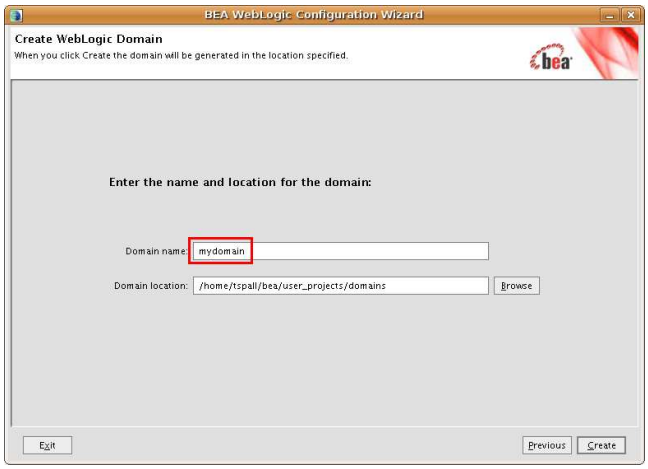
You want to be running in "Development Mode" and use the BEA supplied Sun JDK.



None of the port settings has to be reconfigured.



Specify "mydomain" as domain name and click the "Create" button.



Your new domain will afterwards be located in $WL_HOME/user_projects/domains/mydomain.

B PostgreSQL DB and JDBC driver installation for BEA WebLogic

This appendix describes how to install and configure PostgreSQL 8.3 and it's JDBC drivers for BEA WebLogic 10.0.

B.1 Setting up PostgreSQL

You can download the latest binary packages for PostgreSQL from http://www.postgresql.org/, there is distributions for Linux, Windows and Solaris available. Please refer to the included documentation on how to go through the installation process.

After completing the install you should edit the security and permission settings in the "postgresql.conf" and "pg_hba.conf" files so that the host machine running the BEA WebLogic Server is able to connect to your PostgreSQL DB instance.

```
[e.g. "postgresql.conf"]
...
# - Connection Settings -

listen_addresses = '*'          # what IP address(es) to listen on;
                                # comma-separated list of addresses;
                                # defaults to 'localhost', '*' = all
                                # (change requires restart)
port = 5432                     # (change requires restart)
max_connections = 100           # (change requires restart)
...
```

```
[e.g. "pg_hba.conf"]
...
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD

local   all         postgres                          ident sameuser
# "local" is for Unix domain socket connections only
local   all         all                               ident sameuser
# IPv4 local connections:
host    all         all         127.0.0.1/32          md5
# IPv6 local connections:
host    all         all         ::1/128               md5

host    all         all         192.168.0.0       255.255.255.0     trust
host    all         all         0.0.0.0           0.0.0.0           reject
...
```

B.2 Installing JDBC drivers in WebLogic

You can download the JDBC drivers for PostgreSQL from http://jdbc.postgresql.org/. Select the "JDBC3 Postgresql Driver, Version 8.3-603" version and save the .jar file (postgresql-8.3-603.jdbc3.jar) to your disk.

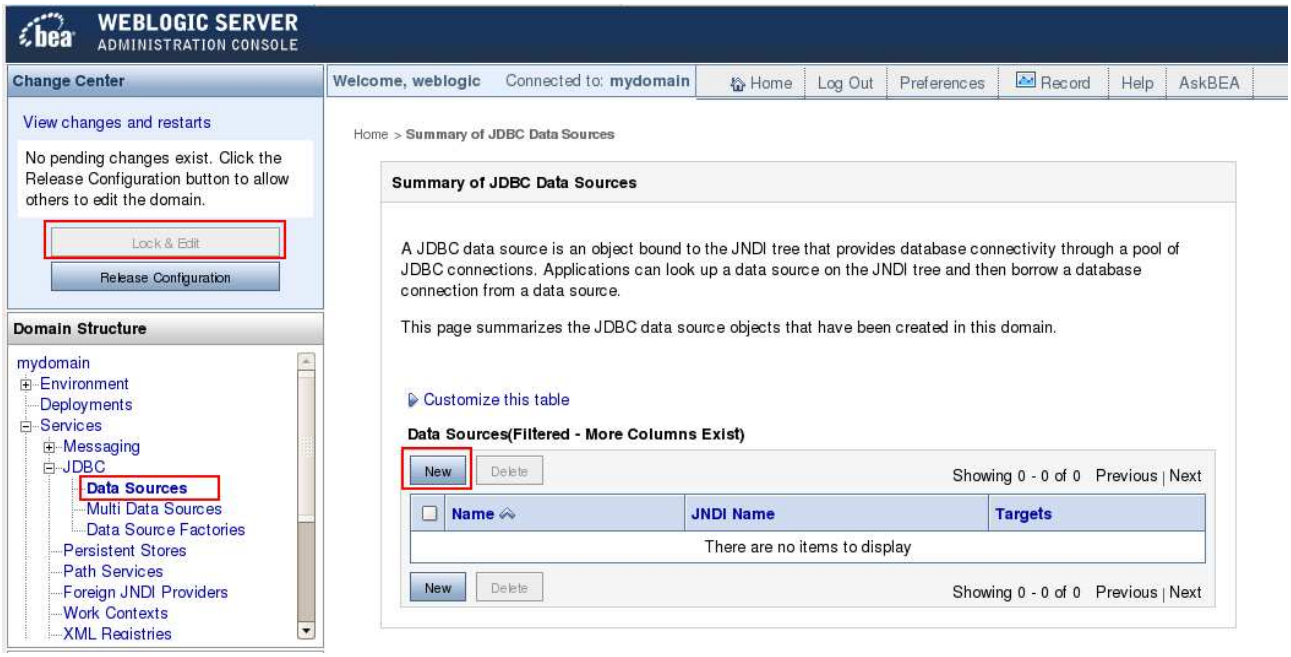
To make the JDBC driver available for WebLogic, copy the .jar file into the directory $WL_HOME/wlserver_10.0/server/lib. Then and add it to server's classpath by appending the "WEBLOGIC_CLASSPATH" environment variable in the file $WL_HOME/wlserver_10.0/common/bin/commEnv.sh.

```
[e.g. "commEnv.sh"]
...
# set up WebLogic Server's class path
WEBLOGIC_CLASSPATH="...
                    ${CLASSPATHSEP}${WL_HOME}/server/lib/postgresql-8.3-603.jdbc3.jar"
export WEBLOGIC_CLASSPATH
...
```

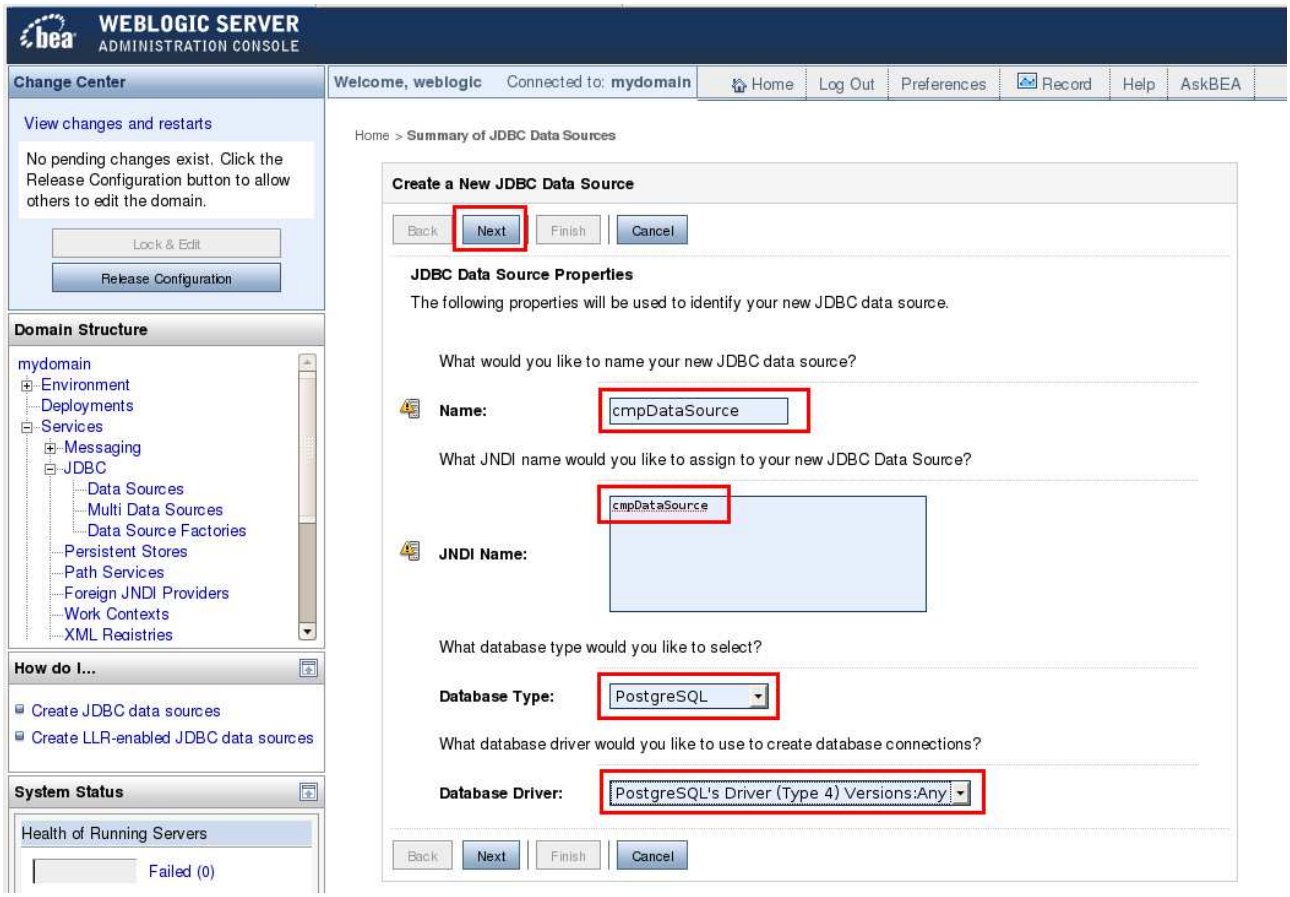
C Configuring a new PostgreSQL JDBC data source in BEA WebLogic

This section describes how to create and correctly configure a new PostgreSQL JDBC Data Source in BEA WebLogic 10.0.

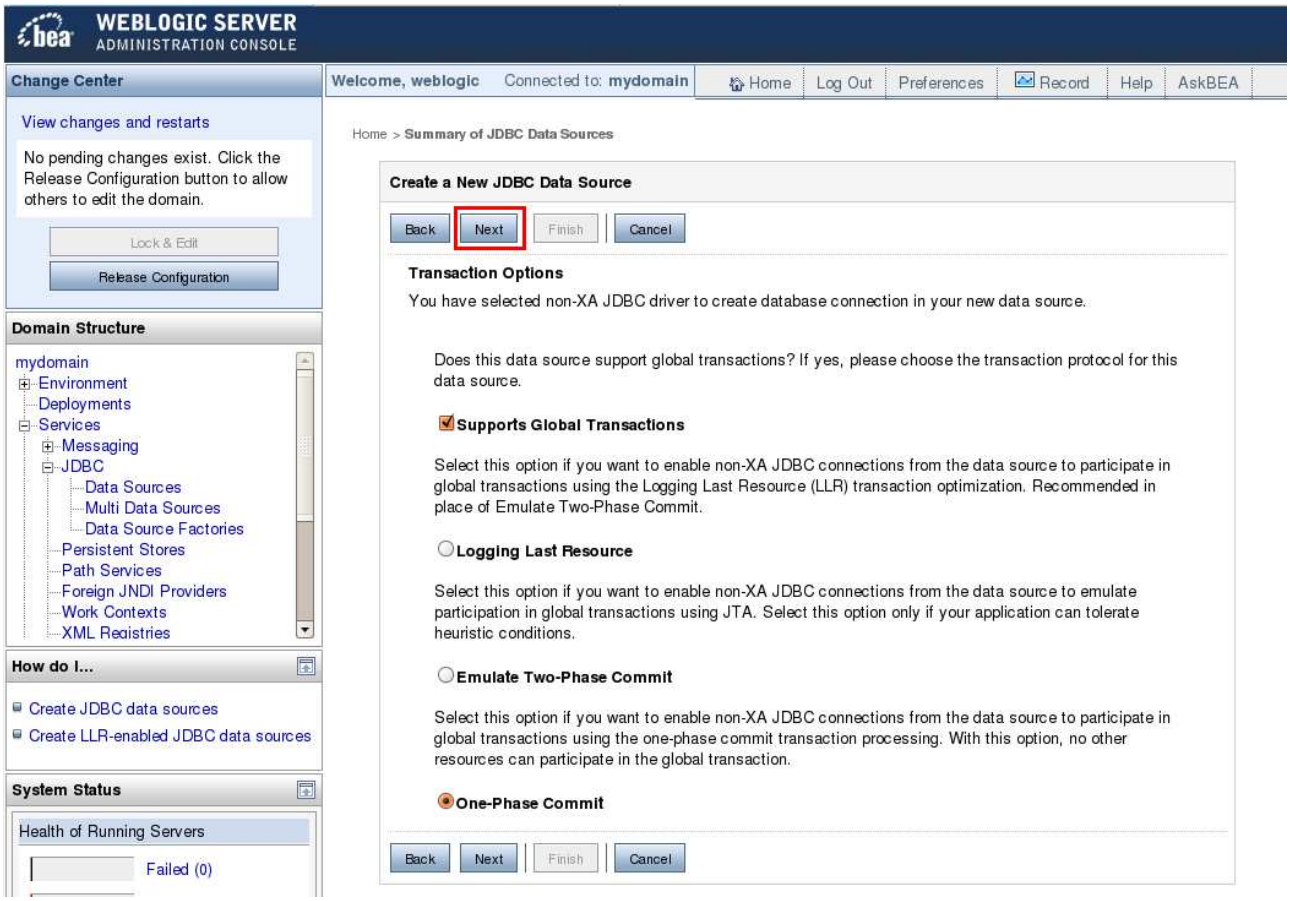
In the WebLogic console menu, open up "myDomain -> Services -> JDBC -> Data Sources", click on "Lock & Edit" and then on "New".



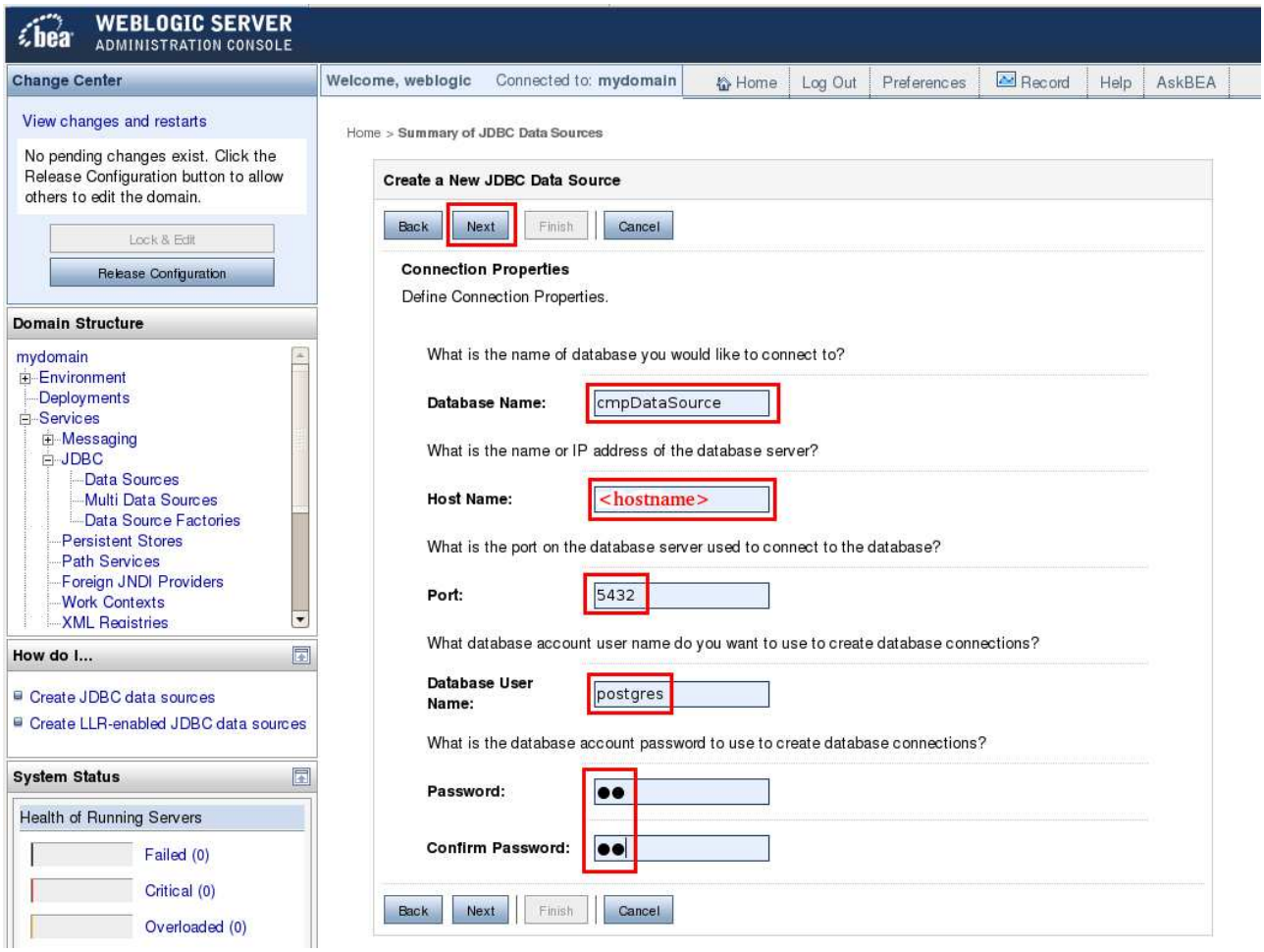
As JNDI name for the new data source choose "cmpDataSource". As database type select "PostgreSQL" from the drop-down box. Then click "Next".



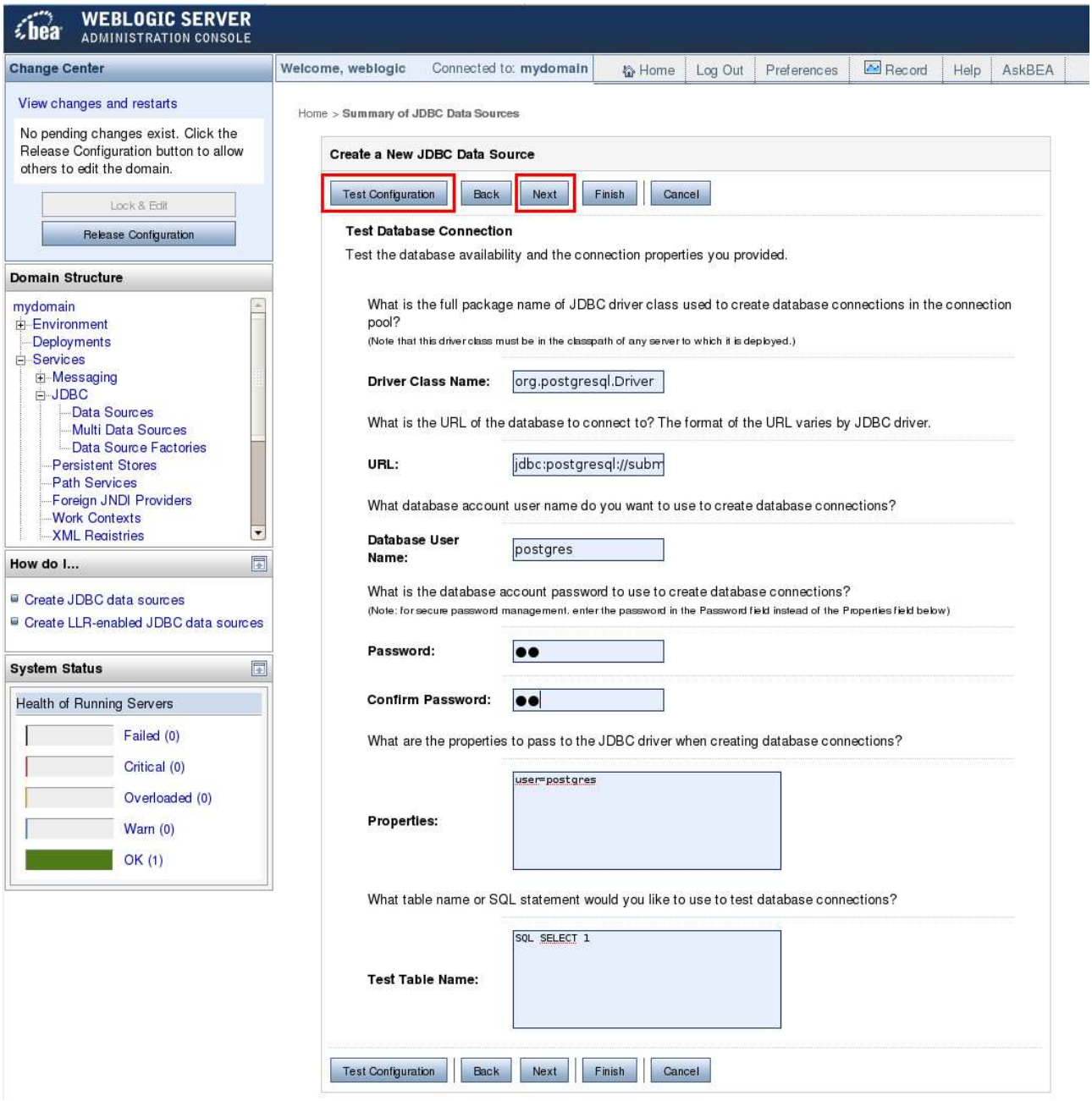
Check that "Support global Transactions" and "One-Phase-Commit" are ticked, then click "Next".



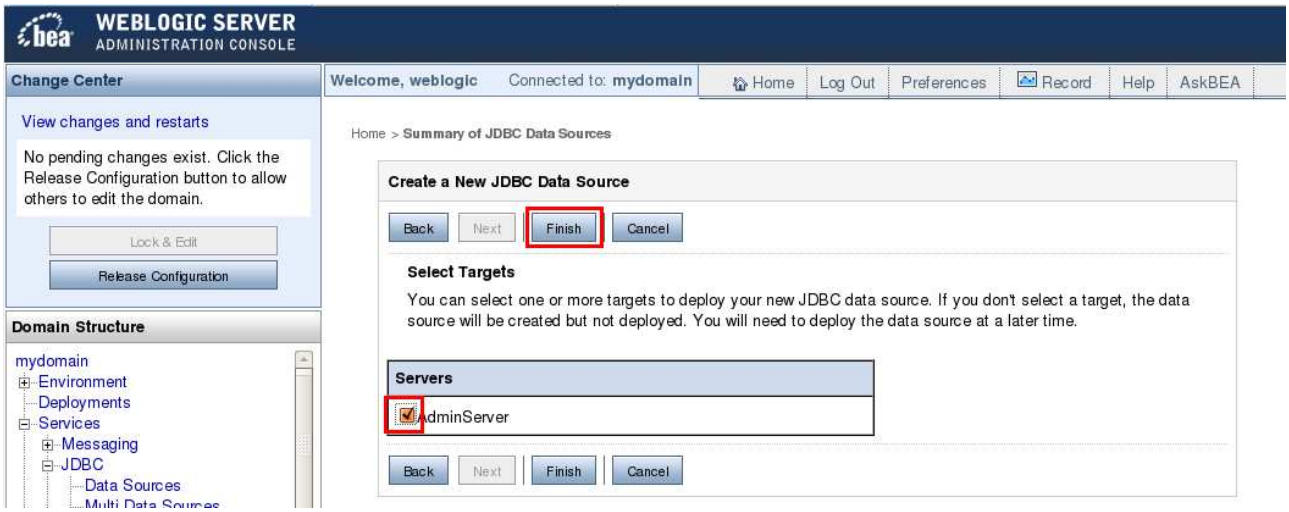
'Database name' should be "cmpDataSource", as 'hostname' please select the IP address or hostname assigned to the machine where your PostgreSQL server is running, the 'port' should be the standard PostgreSQL port 5432 unless you configured it otherwise during the installation of your database server. As 'username' and 'password' please select a valid username-password combination which has been configured for the postgres server and has sufficient access rights (in the example we use the 'postgres' admin account).



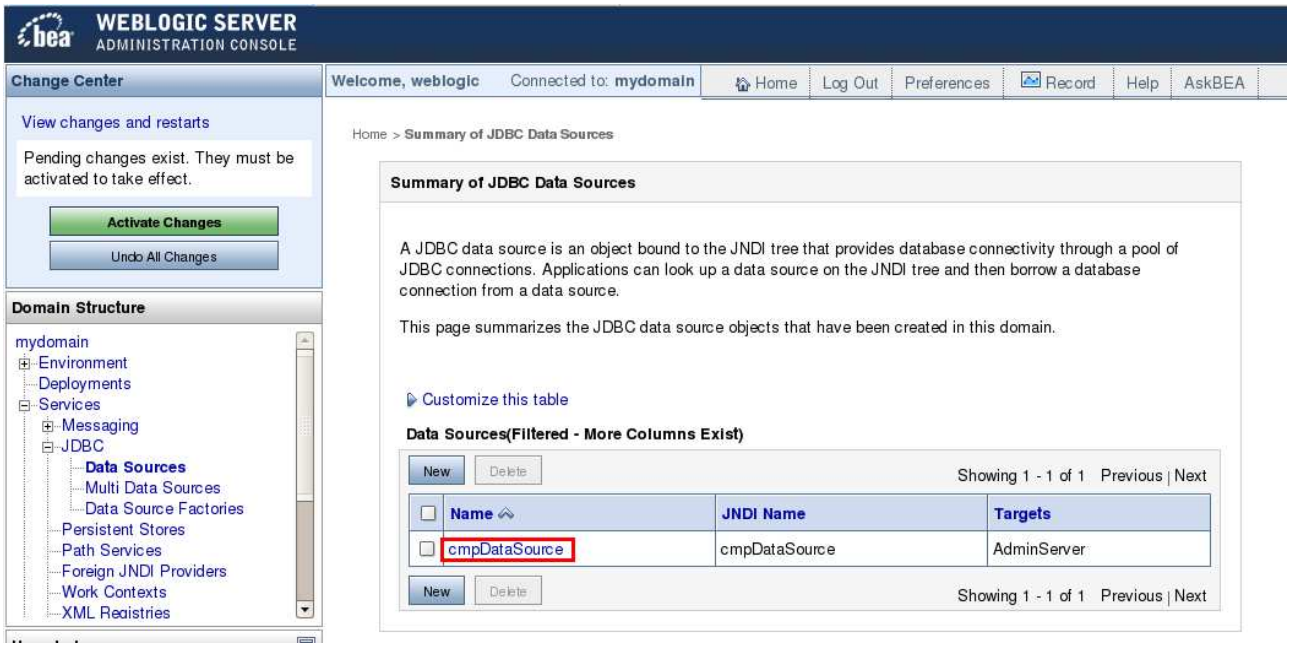
Click on "Test Configuration" and when the test has completed successfully on "Next".



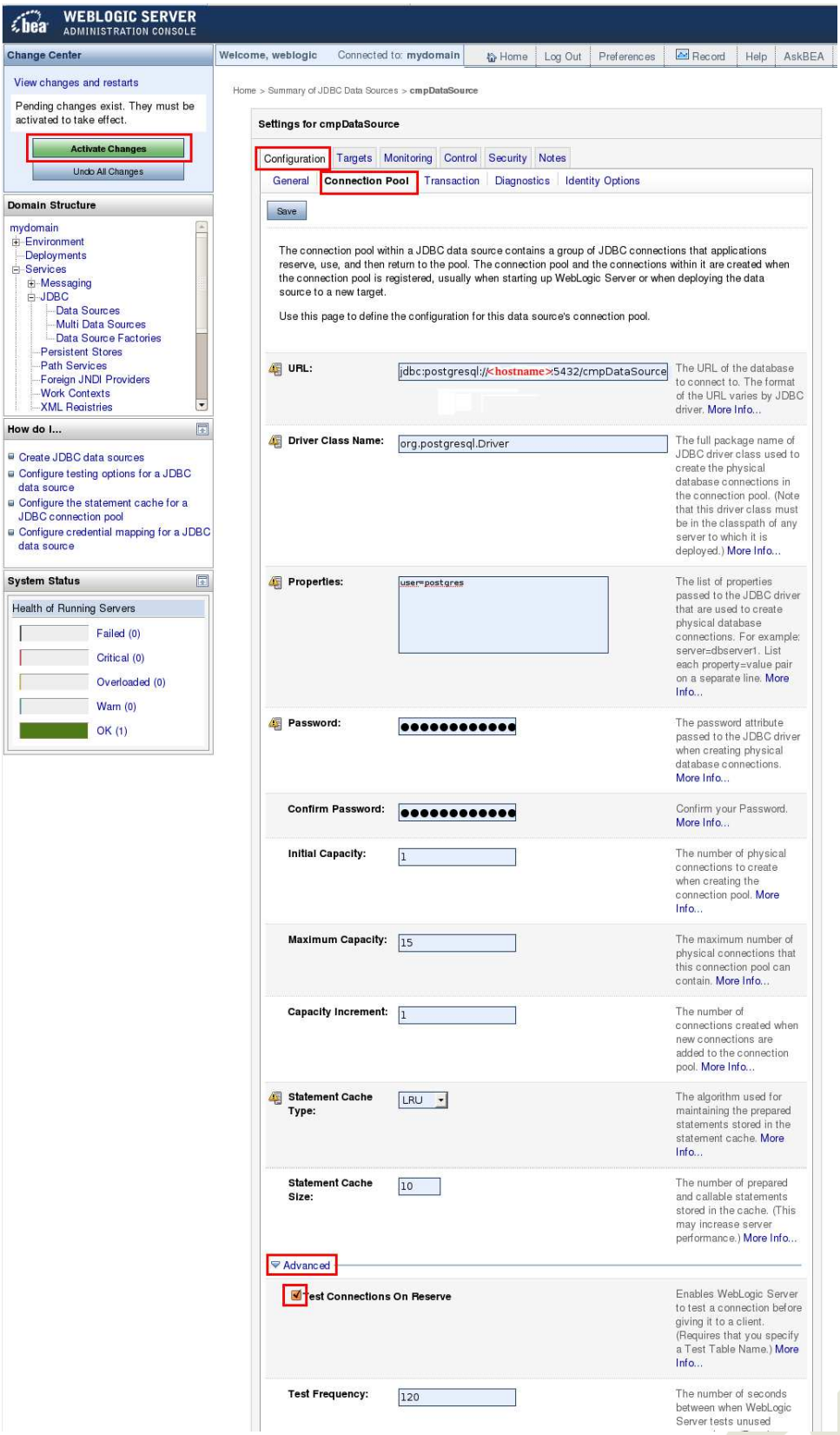
Tick the box besides "AdminServer" and click "Finish"



Now you have the new item "cmpDataSource" listed under the menu "myDomain -> Services -> JDBC -> Data Sources". By clicking on the "cmpDataSource" link you get to the data source's configuration screen.



Click on the "Configuration" tab and then on the "Connection Pool" sub-tab. On the bottom of the configuration list click on "Advanced" to unroll additional configuration options. Then tick the "Test connections on Reserve" box and "Save". Finally click on "Activate Changes" to activate the created datasource in the application server.


WEBLOGIC SERVER
ADMINISTRATION CONSOLE
Change Center
View changes and restarts
Pending changes exist. They must be activated to take effect.
Activate Changes
Undo All Changes
Domain Structure
mydomain
Environment
Deployments
Services
Messaging
JDBC
Data Sources
Multi Data Sources
Data Source Factories
Persistent Stores
Path Services
Foreign JNDI Providers
Work Contexts
XML Registries
How do I...
Create JDBC data sources
Configure testing options for a JDBC data source
Configure the statement cache for a JDBC connection pool
Configure credential mapping for a JDBC data source
System Status
Health of Running Servers
Failed (0)
Critical (0)
Overloaded (0)
Warn (0)
OK (1)
Welcome, weblogic
Connected to: mydomain
Home
Log Out
Preferences
Record
Help
AskBEA
Home > Summary of JDBC Data Sources > cmpDataSource
Settings for cmpDataSource
Configuration
Targets
Monitoring
Control
Security
Notes
General
Connection Pool
Transaction
Diagnostics
Identity Options
Save
The connection pool within a JDBC data source contains a group of JDBC connections that applications reserve, use, and then return to the pool. The connection pool and the connections within it are created when the connection pool is registered, usually when starting up WebLogic Server or when deploying the data source to a new target.
Use this page to define the configuration for this data source's connection pool.
URL:
jdbc:postgresql://<hostname>:5432/cmpDataSource
The URL of the database to connect to. The format of the URL varies by JDBC driver. More Info...
Driver Class Name:
org.postgresql.Driver
The full package name of JDBC driver class used to create the physical database connections in the connection pool. (Note that this driver class must be in the classpath of any server to which it is deployed.) More Info...
Properties:
user=postgres
The list of properties passed to the JDBC driver that are used to create physical database connections. For example: server=dbserver1. List each property=value pair on a separate line. More Info...
Password:
The password attribute passed to the JDBC driver when creating physical database connections. More Info...
Confirm Password:
Confirm your Password. More Info...
Initial Capacity:
1
The number of physical connections to create when creating the connection pool. More Info...
Maximum Capacity:
15
The maximum number of physical connections that this connection pool can contain. More Info...
Capacity Increment:
1
The number of connections created when new connections are added to the connection pool. More Info...
Statement Cache Type:
LRU
The algorithm used for maintaining the prepared statements stored in the statement cache. More Info...
Statement Cache Size:
10
The number of prepared and callable statements stored in the cache. (This may increase server performance.) More Info...
Advanced
Test Connections On Reserve
Enables WebLogic Server to test a connection before giving it to a client. (Requires that you specify a Test Table Name.) More Info...
Test Frequency:
120
The number of seconds between when WebLogic Server tests unused

D Prepopulating the PostgreSQL DB for BEA WebLogic

This appendix describes the steps to put certain data components into a freshly installed PostgreSQL DB so it is ready to be used for a deployment of the J2EE Examples on BEA WebLogic 10.

For the J2EE examples to be deployable in WebLogic you first have to follow some steps to put initial data into your DB. You have to create a new database object within your DB server and the database schema and Account table structure as used by the J2EE examples.

Use the PSQL client tool to connect to the PostgreSQL server.

```
$ /usr/lib/postgresql/8.3/bin>./psql -h <hostname> -U postgres
```

Then run the following SQL commands in the PSQL client to initialize the DB data:

```
postgres=# CREATE DATABASE "cmpDataSource";

postgres=# CREATE SCHEMA account;

postgres=# CREATE TABLE account.account(accountNumber INT PRIMARY KEY, balance FLOAT, accountType
           VARCHAR(25));
```