Structured overlay Networks Application Platform

SNAP

Developer's Manual

Carles Pairot Gavaldà - <carles.pairot@urv.net>
Architecture and Telematic Services Research Group
Universitat Rovira i Virgili
Tarragona
Spain

Web application adaptation guide

Adapting an already existent web application to work with SNAP is just a simple process. There are several kinds of web applications and adaptation modes.

IMPORTANT NOTE: This guide assumes you have downloaded SNAP's source code. You may want to recompile SNAP. For such matter you will require ANT – http://ant.apache.org. Installation directory for SNAP's source code is represented by $SNAP_SRCDIR, whereas SNAP's binary installation is represented by $SNAP_DIR.

STATIC WEB APPLICATION

If your web application is a static one (i.e. includes only static HTML pages, say a home web page), adaptation is just a matter of replacing the default web page for the one included in $SNAP_SRCDIR/templates/index.jsp. Please note that the default original web page must be renamed to orig_index.html. The process is that SNAP will load first the index.jsp page, which will initialize SNAP's environment, and automatically redirecting to orig_index.html, which corresponds to the original static web application index.

SNAP can be used to provide load balancing and failover to static web applications which do not require any kind of database access.

More specifically, what the new index.jsp page does is this:

```
<%@page import="dermi.*, org.planet.snap.*, java.util.*, java.io.*,
                java.net.*;" %>
<%
// Create and register application
Application app = new Application
(InetAddress.getLocalHost().getHostName(), request.getServerPort(),
Application.getSnapAppConfig (application.getRealPath (File.separator)));

application.setAttribute ("snap_webapp", app);

%>

<FRAMESET cols="0%, 100%" FRAMEBORDER="NO" BORDER="0" FRAMESPACING="0">
    <FRAME src="about:blank" SCROLLING="NO" noresize>
    <FRAME src="orig_index.html" noresize>
</FRAMESET>
```

Once this is done you are ready to deploy your static web application onto SNAP. To do so, please refer to the Web Application Deployment section.

DYNAMIC WEB APPLICATION

- Persistence mode: DataSourced Database

In order to adapt a dynamic web application which performs relational database queries working with container provided *DataSources*, we must add the following lines to the application's web.xml descriptor file, indicating the initial load of the SNAP's associated instance:

```
<servlet>
    <servlet-name>startup</servlet-name>
    <servlet-class>org.planet.snap.SnapAppStartup</servlet-class>
    <load-on-startup>0</load-on-startup>
</servlet>
```

By using SNAP's datasource configuration, we guarantee that persistence data is replicated among a determinate number of servers, to guarantee transparent failover and load balancing.

DataSource configuration is to be modified on the application's web.xml descriptor file as well by using the default java:comp/env/jdbc/SnapDS JNDI name. However DataSource name and configuration can be modified by opening the $SNAP_DIR/etc/jetty.xml file. By following such approach not a single line of code is to be changed in order to port the application to SNAP.

```
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<!-- Snap Datasource properties.                                             -->
<!-- + It uses a modified HSQLDB database with replication.                  -->
<!-- + No host is specified, and neither port is.                            -->
<!-- + Only username and password arguments.                                 -->
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<Call name="addService">
  <Arg>
    <New class="org.mortbay.jetty.plus.DefaultDataSourceService">
      <Set name="Name">DataSourceService</Set>
      <Call name="addDataSource">
                  <Arg>jdbc/SnapDS</Arg>
                  <Arg>
                  <New class="org.planet.snap.ds.SnapDataSource">
                        <Set name="Username">sa</Set>
                        <Set name="Password"></Set>
                  </New>
                  </Arg>
                  </Call>
            </New>
     </Arg>
</Call>
```

Once you have changed the DataSource name, you are ready to deploy your web application onto SNAP. To do so, please refer to the Web Application Deployment section.

- Persistence mode: Direct JDBC Connection to Database

If your application does not use DataSourced connections, then you must adapt it only to modify the connection phase so as persistent data is stored and replicated using SNAP's modified HSQLDB bundled version.

The first thing to do is to modify the application's web.xml file, as seen on the previous step (only to load the startup servlet).

By using SNAP's datasource configuration, we guarantee that persistence data is replicated among a determinate number of servers, to guarantee transparent failover and load balancing.

How can a JDBC connection to the underlying SNAP database be obtained? Take a look at the following code snippet:

```
<%@page import="dermi.*, org.planet.p2pcm.*, org.planet.snap.*,
                java.util.*, java.io.*, java.net.*, java.sql.*"
%>

<%
Application app = (Application) application.getAttribute ("snapApp");

if (app == null) {
  throw new NullPointerException ("Application not initialized!");
}

// To obtain a SNAP-compatible JDBC connection
Connection con = app.getConnection ("sa", "");
...
%>
```

First of all, a reference to the underlying SNAP application instance must be obtained. Next, to get a JDBC connection, simply call the getConnection (user, password) method. From now on, the rest of the code remains the same.

Web application deployment guide

In order to be able to deploy any kind of web application onto SNAP, it must be previously approved by the network's administrator.

The ideal case would be to have the application delivered to the administrator, which will in turn sign it and make it available to the SNAP community. Note that if an application is deployed in a SNAP node but it has not been signed by the administrator, it will not work, since this security aspect is checked every time an application is accessed.

However, let us supose the application has been sent to the administrator (YOU ;-)), and you need to deploy it. The first thing to do is to define a file named snap-war.xml, which is to be located in the META-INF directory of the web application.

This file contains SNAP's metadata for this application. More specifically, it specifies the persistence type, the clustering factor (the number of nodes where the application will dynamically be replicated), and database properties. A sample *snap-war.xml* file is shown as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
  <comment>
    SNAP Web-Application descriptor
  </comment>
  <entry key="appName">SNAP TestSuite Application</entry>
  <entry key="appContext">snaptestsuite</entry>
  <entry key="appP2PUrl">p2p://this_entry_is_not_used_now</entry>
  <entry key="persistence">database</entry>
  <entry key="clustering">3</entry>
  <entry key="dbInitialPort">9999</entry>
  <entry key="dbDataPath">/WEB-INF/data/</entry>
  <entry key="dbPassivationThreshold">10</entry>
</properties>
```

This is what the entries mean:

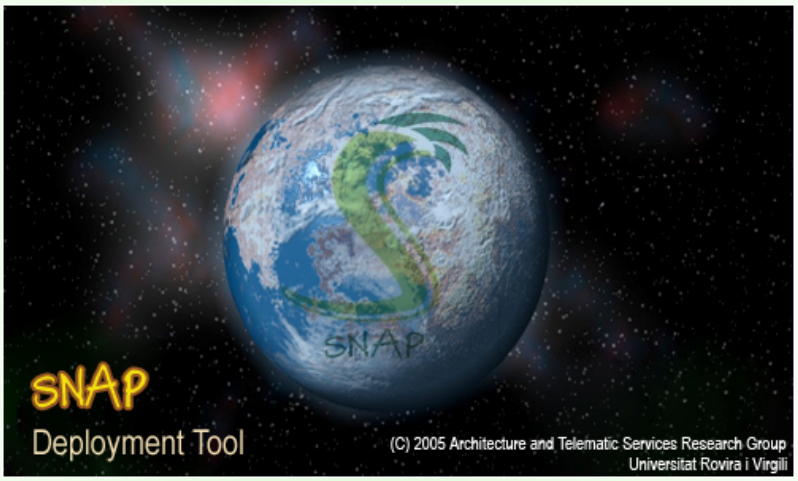
- appName – The application's name
- appContext – The application's context (in Jetty webserver)
- appP2PUrl – Unused entry – to be filled in by SNAPDeployer
- persistence – Persistence type (*database*) or delete the entry for none
- clustering – Clustering factor: the number of nodes where the application will dynamically be replicated
- dbInitialPort – Database initial port (subsequent database activations are to be bound in incremental ports)
- dbDataPath – Path where the database files are to be stored (relative to the application's context)
- dbPassivationThreshold – Minutes of inactivity for database instance passivation (to free up resources)

All entries should be filled in before trying to deploy the web application. Subsequent versions of SNAP will have this process integrated with the deployment tool. However this step is currently required to be done manually.

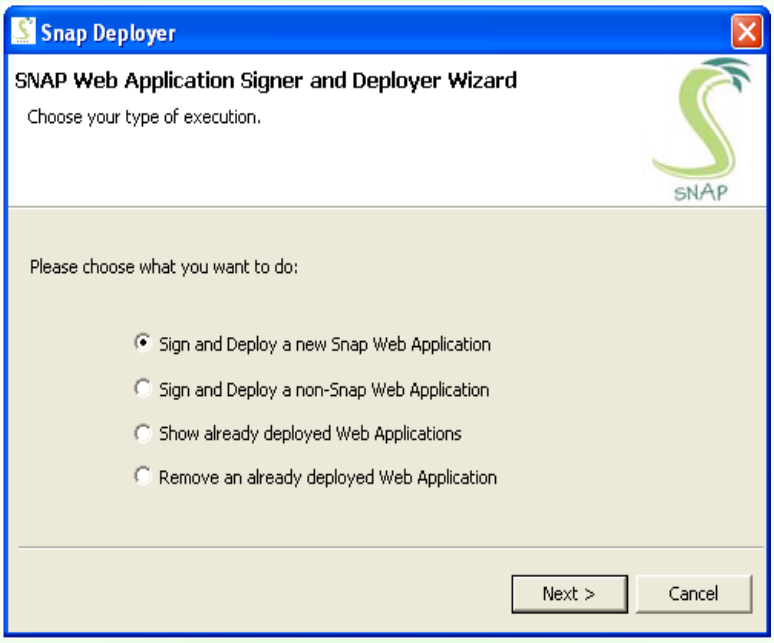
Before making something else, please make sure that the configuration file found in $SNAP_SRCDIR/WEB-INF/dermi-config.xml is correctly updated. Please note that this file is for the source distribution. The binary distribution configuration file is located in $SNAP_DIR/snap/WEB-INF/dermi-config.xml, and you already had to configure it on Installation time (please see SNAP's installation guide). If you prefer, you may copy $SNAP_DIR/snap/WEB-INF/dermi-config.xml over $SNAP_SRCDIR/WEB-INF/dermi-config.xml.

To start the SNAPDeployer tool, you must change directory to $SNAP_SRCDIR, and execute the deployer.bat script.

If everything is OK, the deployer will connect to the network, and show this splash screen:



Next, a step-by-step wizard will guide you through the deployment process. It is important to make a distinction in the options that the wizard presents you with, as seen below:



- Sign and Deploy a new Snap Web Application – This option is for deploying an already .WAR packed application onto SNAP.
- Sign and Deploy a non-Snap Web Application – This option is for deploying a directory containing a web application onto SNAP.
- Show already deployed Web Applications – This option shows information about all active and deployed web applications
- Remove an already deployed Web Application – This option allows undeployment of a SNAP web application.

Next deployment step generates an administrator public/private key pair (if it is the first time), or reuses a keystore.rsa file which contains the administrators public/private key pair (you should keep this file in a safe place).

Finally, and after asking about application's metadata (it is specially important to assign a correct p2p:// identifier, as this will be used to locate the application via SNAP's decentralized application locator), the application will be signed, repacked and uploaded to the decentralized SNAP network.