

Procedures for Installing SigPath with FastObjects

Date: 3/12/2004 6:07:48 PM

Introduction

This document describes the general procedures for building and deploying SigPath from a source code distribution. SigPath requires a JDO compliant database such as FastObjects or Kodo, and also a servlet container such as Tomcat. This document describes deploying to a machine using FastObjects as the backend database and Apache Tomcat for the web application. Although these procedures are tailored to Windows 2000/XP installations, installing to a Unix based system, including Max OSX, should be quite similar.

Required Software Packages

The following software packages are required in order to fully install SigPath onto a local machine.

CVS Client¹

The latest SigPath code is available using CVS. A command line interface and general documentation about CVS is available at <http://www.cvshome.org/>. Other GUI based clients are also available at that site. Good CVS clients are embedded into most IDEs such as IntelliJ IDEA and NetBeans.

Java Development Kit (JDK)

The Java Development Kit is required to compile and run SigPath. Download (<http://java.sun.com/>) and install the Java SDK into separate directory. SigPath has been tested with version 1.4.x and 1.5.0beta 1 of Java.

Apache Ant

Download (<http://ant.apache.org/>) and install Apache Ant into separate directory. SigPath now requires version 1.6 of Ant, or greater. This version is needed to support the onerror attribute on taskdef elements.

JUnit

Download (<http://www.junit.org/>) JUnit and copy junit.jar into <ANT_HOME>/lib. Alternatively, a copy of junit.jar is available in <sigpath>/lib.

Apache Tomcat

Download (<http://jakarta.apache.org/tomcat/>) and install Apache Tomcat into separate directory. At the moment, SigPath has been known to work with Tomcat versions 4.0.6 through 5.1.9. Version 5 or better is recommended.

¹ Note that public CVS access is not yet available at this time.

FastObjects

Download (<http://www.fastobjects.com/>) and install FastObjects t7 and apply patch 9.0.7.185 into separate directory. The required jar files for FastObjects need to be copied into <sigpath>/lib. Only the JDO files are required. The ODMG versions are no longer needed or supported.

Configuring External Resources

User Environment

This section describes the environment variables required for SigPath. Most are described in the installation procedures above. On Windows based machines, these can be set by Right click on “My Computer” and select “Properties”. Select the “Advanced” tab and then click on “Environment Variables”. Typically, these variables should be placed under the “user” environment, and not the system.

JAVA_HOME

Used to locate Java Installation

Variable Value: <JDK installation directory>

Example: 'C:\java\jdk1.3.1_09'

ANT_HOME

Used to locate Ant Installation

Variable Value: <apache ant installation directory>

Example: 'C:\java\apache-ant-1.6.1'

FASTOBJECTS_HOME

Used to locate FastObjects Installation

Variable Value: <FastObjects installation directory>

Example: 'C:\Program Files\FastObjects_t7_9.0'

CVSROOT²

Specifies where CVS should look for source code.

Variable Value: ':pserver:<username>@icbtools.med.cornell.edu:/home/cvsroot'

Example: ':pserver:michael@icbtools.med.cornell.edu:/home/cvsroot'

TOMCAT_DEPLOY³

Specifies webapps directory to be used by SigPath

Variable Value #6: <Tomcat webapps directory for sigpath>

Example: 'C:\Program Files\Apache Software Foundation\Tomcat 5.0\webapps\sigpath'

² This is not required if you pass the CVSROOT to cvs via the command line option “-d”.

³ This will probably be moved to <sigpath>/config/sigpath.properties eventually.

PATH

This specifies the location for tools and commands such as ant and ptserver. This will most certainly contain more than what is described here.

Example:

```
'%FASTOBJECTS_HOME%\runtime\bin;%FASTOBJECTS_HOME%\bin;%JAVA_HOME%\bin;%ANT_HOME%\bin;C:\Program Files\Apache Software Foundation\Tomcat 5.0'
```

The following variables no longer need to be set:

- **CLASSPATH**
- **DATABASE**
- **PTINIFILE⁴**

FastObjects

Configuration

Server (ptserver.cfg)

Configuration of the FastObjects server is handled via the file <FASTOBJECTS_HOME>/runtime/bin/ptserver.cfg. Basically, you need to set up entries for the SigPath database (sigpath_base) and the corresponding schemata (sigpath_dict). Typically, you'll have multiple entries for development, production release, beta, etc. A suggested configuration would look something like:

```
[servers\ptserver]
# Activate server output
# verboseMode=2

[databases\base]
# Replace <FastObjects> with your FastObjects installation directory
# name=<FastObjects>/examples/basics/base
# name=<FastObjects>/examples/advanced/base

#
# Sigpath production release
#
[databases\sigpath_prod]
name=c:/sigpath_db/sigpath_prod/sigpath_base

[schemata\sigpath_prod_dict]
name=c:/sigpath_db/sigpath_prod/sigpath_dict

#
# Sigpath development/test
```

⁴ This only needs to be set if you don't specify the initialization file when you start the fastobjects server.

```
#
[databases\sigpath_dev]
name=c:/sigpath_db/sigpath_dev/sigpath_base

[schemata\sigpath_dev_dict]
name=c:/sigpath_db/sigpath_dev/sigpath_dict
```

Client (poet.cfg)

No modifications should be required for poet.cfg.

Starting the server

The FastObjects server can be started via the ptserver command. To start the server, issue the following command:

```
<FASTOBJECTS_HOME>/runtime/bin/ptserver -config
<FASTOBJECTS_HOME>runtime/bin/ptserver.cfg.
```

Alternatively, if you are on a Windows machine, you could start the server with the icon located on the “start” menu. Be sure to check the output to make sure the correct configuration file is being used.

Stopping the Server

It is necessary to restart the server if the configuration file is changed. This is done using the command “ptsu”. This is a relatively simple command interface, but documentation on how to do this is available with the FastObjects installation.

SigPath

This section describes how to build, test and deploy SigPath. Commands described in this section assume that the user is in the “sigpath” directory after the code has been checked out from CVS or downloaded from the SigPath website and that all references to directories are relative to the “sigpath” directory. It is also assumed that the FastObjects server that you will use is configured and running at this point.

SigPath Source code

The latest version of the SigPath source code is available in CVS.

Getting a new copy

Create a new directory and cd into it.

```
cvs -d :pserver:<username>@icbtools.med.cornell.edu:/home/cvsroot login5
cvs -d :pserver:<username>@icbtools.med.cornell.edu:/home/cvsroot co -r sp-jdo
sigpath
```

Updating an existing copy

Switch into the top level of your local sigpath directory

```
cvs -d :pserver:<username>@goya.med.cornell.edu:/home/cvsroot update -Pd
```

⁵ Typically, you’ll only need to do this once, but it never hurts to do it again

Configuration

SigPath assumes that configuration data is located in *config/sigpath.properties*. This file does not exist in cvs, and is created using based on the configuration properties set in the ant build script *build/build_config.xml*.

Each SigPath configuration has a separate section, typically named something like “config_benchmark_fastobjects”, in this file. A sample configuration is listed below. It is often best to copy a known good existing section and creating a new one for yourself.

Sample configuration

Typically, the only changes that are necessary are for the host information that is running the FastObjects server and the FastObjects client license key. A sample configuration for running some benchmark tests are listed below:

```
<target name="config_benchmark_fastobjects" depends="init"
description="Configure fastobjects database to run benchmarks">
  <antcall target="reset_config_files"/>
  <antcall target="copy_config_files_fastobjects"/>
  <antcall target="copy_struts_config"/>
  <antcall target="config_db">
    <param name="edu.mssm.crover.sigpath.JdoImplementation"
value="FastObjects"/>
    <param name="javax.jdo.PersistenceManagerFactoryClass"
value="com.poet.jdo.PersistenceManagerFactories"/>
    <param name="javax.jdo.option.ConnectionDriverName"
value="" />
    <param name="javax.jdo.option.ConnectionUserName"
value="" />
    <param name="javax.jdo.option.ConnectionPassword"
value="" />
    <param name="javax.jdo.option.ConnectionURL"
value="FastObjects://localhost/sigpath_dev"/>
    <param name="com.fastobjects.dictionary.name"
value="sigpath_dev_dict"/>
    <param name="com.fastobjects.database.name"
value="sigpath_dev"/>
    <param name="com.fastobjects.database.server"
value="localhost"/>
    <param name="com.fastobjects.database.physical"
value="c:/sigpath_db/sigpath_dev"/>
    <param name="com.fastobjects.license" value=" ZAFLA-XXXXX-
YYYYY-HPZZZ"/>
    <param name="javax.jdo.Extension" value="FOJDO"/>
    <param name="javax.jdo.option.MinPool" value="0"/>
    <param name="javax.jdo.option.MaxPool" value="0"/>
  </antcall>
  <antcall target="config_fts">
    <param name="ftsMethod" value="{foFTS}"/>
    <param name="oracleDriver" value="{winOraDriver}"/>
    <param name="oracleURL" value="{unix-cornell-OraURL}"/>
    <param name="oracleSuffix" value="BBC"/>
  </antcall>
  <antcall target="config_others">
    <param name="isPrimarySpidGenerator" value="true"/>
  </antcall>
</target>
```

```

        <param name="spidRangeSize" value="100"/>
        <param name="spidGeneratorURL"
value="http://localhost:8080/sigpath/getSpidRange.action?ID=SP-ALPHA-
ICB"/>
        <param name="graphTechnology" value="None"/> <!-- can be
YFiles, Tomsawyer or None. -->
    </antcall>
</target>

```

You should copy this sample configuration to `build_config.xml` and rename the target something else. The new name should allow you to identify the target. Each developer should have his/her own config target. This is a trick that we use to push back an individual configuration to the CVS repository and keep everybody's configuration distinct and not overwriting each other.

Saving the configuration

In order to create the appropriate configuration files, execute the following command, using the configuration you set up above:

```
ant -f build/build_config.xml config_benchmark_fastobjects
```

Starting with a “clean” setup

It is often best to start with a “clean” copy of all the code, especially if you have been making a number of changes to the code and/or configuration. This will force any old classes to be removed, and any other “old” data to be removed. To do this, execute the following command:

```
ant -f build/build.xml clean
```

Creating the initial schema

In order to run the benchmarks, a schema must be registered with FastObjects. The schema can be created by executing the following command;

```
ant -f build/build.xml jdo_createschema_fastobjects
```

Typically, you’ll only need to do this once. Also, it should be noted that this step will delete any existing data stored previously in your configuration.

Building the Project

The following section describes the steps to build SigPath. It should be noted that all the steps in this section are executed automatically when either testing or deploying SigPath as described below. Additionally, each task automatically executes it’s dependant tasks, so for example the “boot” target, will call the “enhance” target, which will in turn call the “compile” target. These are listed here mainly for illustration purposes.

Compiling

In order to compile the java code for SigPath, execute the following command:

```
ant -f build/build.xml compile
```

Any errors reported in this step must be fixed before proceeding. It is recommended that any warnings be treated as errors and also fixed.

Enhancing the classes

In order to make the Java classes “persistence-capable”, the compiled classes must be modified to support this. To enhance the classes, execute the following command:

```
ant -f build/build.xml jdo_enhance
```

Loading sample “bootstrap” data

SigPath requires some initial “bootstrap” data such as user definitions and sample data used for the unit tests. The boot target is invoked by executing the following command:

```
ant -f build/build.xml boot
```

Testing the configuration

It is strongly recommended that the SigPath version be tested before deployment, particularly whenever changes have been made. SigPath includes a number of unit tests, written using the JUnit framework. Source code for the unit tests can be found in the *src/edu/mssm/crover/sigpath/test* directory. The tests can be executed by issuing the following command:

```
ant -f build/build.xml test
```

Test results are stored in a directory called “test-results”. For each suite of tests, there should be a file named something like “TEST-edu.mssm.crover.sigpath.test.XXX”. There is a plain text version and an xml version of each file. The details contained in each version are the same, only the formatting differs. The test results file includes times for each individual test as well as total time for all the tests.

Initializing Background Data

This section describes how to set up the background and submitted data for SigPath to use.

Details TBD.

Deploying the SigPath web application

The build scripts will package and deploy sigpath to a Tomcat installation via a Web Application Archive (.war) file called sigpath.war. It may be necessary to restart the Tomcat engine after deploying SigPath.

“Automatic” deployment

It is assumed that Tomcat is not running when the sigpath deploy target is invoked. This is basically because the deploy target will delete the previous installation, and this may fail if the files are in use.

Execute the following command to build and deploy the SigPath web application:

```
ant -f build/build.xml deploy
```

“Manual” deployment

It may sometimes be desirable to build the SigPath web archive, and manually copy it to the Tomcat web application directory, perhaps because tomcat is running on another host. If you want to simply build the .war file, execute the following command:

```
ant -f build/build.xml war
```

The resulting sigpath.war file will be placed into the <sigpath> directory and can be copied to any number of tomcat installations.

Verifying the SigPath web application

Start the Tomcat server by executing <Tomcat_Installation>/bin/startup.bat.

Open a web browser and type in the following URL: <http://localhost:8080/sigpath/>. You should then see the SigPath main menu appear in your browser. You should also run a set of tests against the deployment. Details TBD.

Viewing the “raw” SigPath database

You may use the FastObjects “developer” tool to view objects stored in the FastObjects database. The FastObjects server must be running on the host the database is physically located on.

Assuming your database configuration is called “sigpath_dev” and the server is running on the local machine, use “localhost” in the host field, and “sigpath_dev” in the database field. The developer should open and display statistics about your database if everything is working properly. You can then browse the various objects in the database.