Installation and Use of Holyoke Server (hokserver)

Robert Fischer

February 1, 2009

1 Introduction

The Holyoke Framework (holyokefw) is a library used to build two-tier client-server application that operate over the Internet. The server segment of the application is a simple PostgreSQL database, sitting behind an stunnel SSL tunnel. The Holyoke Server package (hokserver) is used to administer applications and users to be served by a particular Holyoke Server instance.

2 Configurations

One of the goals of the server is to build configurations for supported application programs. A configuration is simply a directory tree of files that is made accessible to a program at runtime — either by embedding the files in a launcher program, or by transporting them across the network by a configuration server.

Applications rely on not just one configuration, but rather a series of configurations in decreasing priority order. Files found in a higher-priority configuration mask files in lower-priority configurations. In addition to a number of configurations that might be loaded at runtime, each application has built-in a *default configuration*, which provides paramters not provided by any other configurations.

In most cases, higher-priority configuration files mask lower-priority files. However, Java properties files (ending in `.properties`) are somewhat special. In that case, the Java properties files are concatenated together, with the lowest-priority file first. The effect is that property settings found in higher-priority properties files override proprety settings in lower-priority properties files.

Holyoke Server allows configurations to be set at many different "levels," which allows the system administrator flexibility and efficiency in determining the final configuration for an application.

For example, Holyoke Server allows configurations on a per-application and per-user basis. It might set many basic defaults in the per-application configuration file — including the database server to use (which will presumeably be the computer on which Holyoke Server is running). Each per-customer configuration does not need to repeat all these parameters; it just needs to set up a database username and password.

More formally, Holyoke Server provides for the following configurations (lowest priority first):

apps Base configuration for each supported application (eg, offstagearts); sets things that are the same for all users.

app_vers Special configuration for a particular version of each application (eg, offstagearts v1.9). In many cases, this will be blank.

custs Per-customer configuration. A customer corresponds to one database user; thus, per-customer configuration will normally include a database username and password, as well as any SSL keys.

app_custs Configuration for a particular customer using a particular application. At minimum, it contains the database corresponding to this usage.

users Configuration for each *user* within a customer. A user is a username and password that validates the dynamic configuration server. This will usually be blank.

2.1 Prerequisites

In order to run Holyoke Server, you need the following installed:

1. PostgreSQL version 8
2. Java JDK version 1.5 or higher. NOTE: Use the Sun JDK; the free software JDKs are incompatible and will not work.
3. Maven (if you wish to compile). Try on Fedora: `yum install maven2`

3 Downloading and Compiling

If you already have a `hokserver.tar.gz` file, you may skip this section. Otherwise:

1. `mkdir $HOME/.m2`
2. Create `$HOME/.m2/settings.xml` as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<settings>
  <localRepository/>
    <profiles>
        <profile>
            <id>defaultProfile</id>
            <activation>
                <activeByDefault>true</activeByDefault>
            </activation>

            <repositories>
      <repository>
          <id>offstagearts-releases</id>
  <name>OffstageArts Releases</name>
  <url>http://citibob.net/mvnrepositories/releases</url>
  <layout>default</layout>
      </repository>
      <repository>
```

```
            <id>offstagearts.org 3d Party</id>
      <name>OffstageArts-related Third-Party artifacts</name>
      <url>http://citibob.net/mvnrepositories/thirdparty</url>
      <layout>default</layout>
          </repository>
                </repositories>
            </profile>
        </profiles>

    </settings>
```

3. Check out the source code for Holyoke Server:

   ```
   svn co http://svn.berlios.de/svnroot/repos/holyokefw/trunk/hokserver
   ```

4. Compile: `mvn install -P netbeans-public`

5. Create `target/hokserver.tar.gz` for the next step:

   ```
   sh ./mkdist.sh
   ```

4 Setting up Holyoke Server

4.1 Unpack

Untar the distribution file `hokserver.tar.gz` into your favorit place (eg: into `/opt`). You might wish to make soft links of `/opt/hokserver/bin/*` into `/usr/local/bin`.

4.2 Configure

Holyoke Server needs a directory to write application and user information into. By default, these files will go in `$HOME/.hokserver`. If you wish them to go some other place, you need to set the environment variable `HOKSERVER_HOME`. Whatever you choose, the rest of this document will assume that `$HOKSERVER_HOME` expands to that directory. Now issue (modifying as needed):

```
cd /opt/hokserver
mkdir -p $HOKSERVER_HOME
cp -r baseconfig $HOKSERVER_HOME/config
```

Now edit the file `HOKSERVER_HOME/config/app.properties`. Holyoke Server creates a database in which it keeps track of its users and applications. That database should be specified by the properties *db.database*, *db.user* and *db.password*. Note that neither the database nor the user needs to be manually created; however, if you do create them manually, then Holyoke Server will use those.

Holyoke Server also needs to create new databases, for which it needs superuser priviledges. The properties *admin.db.user* and *admin.db.password* should indicate the super user account on the database server.

4.3 Intialize

Run the command `bin/hsadmin init`. This will create the Holyoke Server database and various files and directories inside `HOKSERVER_HOME`.

4.4 Add an Application

You can now add an application that will be administered by Holyoke Server. For example (all on one line):

```
bin/hsadmin add appvers offstagearts 1.9 \
    http://offstagearts.org/releases/offstagearts/\
        offstagearts_launch-1.9.jnlp \
    true
```

Once an application is added, you need to add an application configuration. In this example, `OFFSTAGEARTS_SRC` refers to the root directory of the *OffstageArts* project, available via svn.

```
mkdir -p $HOKSERVER_HOME/data/configs/apps
cp -r $OFFSTAGEARTS_SRC/baseconfig \
    $HOKSERVER_HOKE/data/configs/apps/offstagearts
```

Alternatively, you can use the following commands if you do not have *OffstageArts* source tree already:

```
mkdir -p $HOKSERVER_HOME/data/configs/apps
svn export \
    http://svn.berlios.de/svnroot/repos/offstagearts/\
        trunk/offstagearts/baseconfig \
    $HOKSERVER_HOME/data/configs/apps/offstagearts
```

You may now edit the files just copied, as appropriate.

4.5 Add a Customer

You are now ready to add a customer to the application:

```
bin/hsadmin add appcust offstagearts ballettheatre 1.9
```

4.6 Add Users (optional)

You may also add users to the app/customer combination, if you are using the dynamic configuration server:

```
bin/hsadmin add user offstagearts ballettheatre bob password
```

5 Editing Configurations

Once configurations have been added to the Holyoke Server using the above methods, one might wish to customize the configurations that have been generated. Configurations are all found in the various subfolders of `$HOKSERVER_HOME/data/configs`, where the subfolder name corresponds to the database table name (*apps, app_vers, custs, app_custs, users*).

The administrator may create new directories — and new files within those directories — in order to add configurations. For example:

- Configuration for *OffstageArts* version 1.9 only: `$HOKSERVER_HOME/data/app_vers/offstagearts`
- Configuration for *OffstageArts* in general: `$HOKSERVER_HOME/data/apps/offstagearts`

6 Making a Launcher

It is now easy to make a custom launcher for your customers:

```
bin/hsadmin mklauncher offstagearts ballettheatre my-launcher.jar password
```

The password is optional. If you wish to see the configuration that is created inside the launcher, you can do:

```
bin/hsadmin dump my-launcher.jar
```

It is preferable that launchers be debugged without password.

7 Running the SSL Server

Now that applications and users have been created on the server, one would like to connect applications to it. In the case that some applications look to connect with `db.ssl = true`, then the SSL tunnel needs to be running. There are three ways to do this: the Java SSL tunnel, the prototype stunnel, and the final stunnel

7.1 Java SSL Tunnel

On can launch the Java SSL tunnel using `bin/hsssl`. This is for test purposes only; however, once launched, all access to the local database server should work, using the cryptographic keys that were determined while setting up new customers.

7.2 Prototype SSL Tunnel

The command `./runstunnel.sh` sets up a prototype `stunnel.conf` file and runs stunnel on it. This should work as well; however, it is not set up with the operating system's services.

7.3 Final Stunnel

With the prototype `stunnel.conf` file built, a system administrator should be able to use that to set up a permanent system-wide stunnel. Details depend on the operating system.

8 Building a Dynamic Configuration Server

A dynamic configuration server can be easily built, using the stored procedure: *r_configs_get(rsname, appname, custname, username, password_md5)*. For example:

```
select r_configs_get('rs1', 'offstagearts','ballettheatre',
    'bob',md5('password'));
fetch all in rs1;
```

If the application/customer/user/password combination is correct, then this SQL will return one or two rows of data; the first row should be used. If the combination is not correct, then nothing will be returned.

The following columns (if non-null) should be concatenated to produce a configuration output stream to be sent to the client: *users_config, app_custs_config, custs_config, app_vers_config, apps_config*. This ordering is from highest priority configuration to lowest priority. Files and properties in the higher priority configurations override items in the lower priority configurations.