MavenConfiguration

Maven Configuration Notes

This page provides some notes on setting up Maven. Once set up, most of this will not need to be changed.

Configuration files

Maven has some configuration files which need to be configured for each installation. These files define local settings, such as login names and passwords for the user, and JDK installation locations for the host.

The settings files are located in either of:

- A user's home directory: ${user.home}/.m2/settings.xml (preferred)
- The Maven install directory: $M2_HOME/conf/settings.xml (don't use this for login names and passwords!)

Defining GPG settings

In order to sign releases, you must have a working installation of GPG on your system.

Although it is possible to provide the GPG properties as command-line parameters, it's a lot easier to define them in the local user settings file. Also, by using profiles, it's possible to have multiple GPG settings. For example, you may wish to set up a dummy signing key which can be used for checking local deployments, and you may wish to keep some private keys on removable media, e.g. a USB stick. If you omit the password, it will be prompted for by Maven.

The following settings.xml excerpt shows how to do this:

```
<?xml version="1.0" encoding="UTF-8"?>
<settings>
...
  <profiles>
    <profile>
      <!-- Dummy signing key for testing deploy signing, so we don't care if the password is public -->
      <id>keyTest</id>
      <properties>
        <gpg.keyname>Deploy Test User</gpg.keyname>
        <gpg.passphrase>Deploy Test User Password</gpg.passphrase>
      </properties>
    </profile>
    <profile>
      <!-- Real signing key. Use the hex key id. Don't define the password. -->
      <id>keyReal</id>
      <properties>
        <gpg.keyname>abcdefgh</gpg.keyname>
        <!-- In this case, the secret key is stored on a USB stick -->
        <gpg.homedir>U:/GPG_home</gpg.homedir>
      </properties>
    </profile>
...
</settings>
```

Defining a master password

Recent versions of Maven (e.g. 2.1.0) support password encryption - see http://maven.apache.org/guides/mini/guide-encryption.html The master password should be stored on a USB stick if your host system is not secure. *Note* The plain-text master password does not seem to be needed, once it has been used to create the encrypted master password.

Create a {`${user.home}/.m2/settings-security.xml`} file, for example:

```
<settingsSecurity>
  <relocation>/USB/settings-security.xml</relocation>
</settingsSecurity>
```

This should point to the file containing the master password which is stored elsewhere:

```
<settingsSecurity>
    <master>{Encrypted master password goes here}</master>
</settingsSecurity>
```

If you are sure that your system is secure, then the master password can be stored directly in the `settings-security.xml` file

Once you have set up the master password, you can use it to encrypt any other passwords that you want to include in `settings.xml`

Defining server passwords for uploading files

The `<servers>` section in the `settings.xml` file should contain details for all the servers that Maven needs to contact. The ids for these servers are defined in the `<distributionManagemenr>` section of the POM and are:

- apache.website - for site uploads
- apache.releases.https - Nexus release uploads
- apache.snapshots.https - Nexus snapshot uploads

Note that the previous version (4.0) of the HttpComponents parent POM used different server names for releases and snapshots. These were `apache.releases` and `apache.releases`. These did not use Nexus, and should no longer be used.

Here is an example setup:

```
...
  <servers>
    <server>
      <id>apache.website</id>
      <username><!-- your apache id --></username>
      <passphrase><!-- your GPG pass phrase --></passphrase>
      <directoryPermissions>775</directoryPermissions>
      <filePermissions>644</filePermissions>
    </server>
    <!-- To publish a release -->
    <server>
      <id>apache.releases.https</id>
      <username><!-- your apache id --></username>
      <password>{encryptedpassword}</password>
    </server>
    <!-- To publish a snapshot -->
    <server>
      <id>apache.snapshots.https</id>
      <username><!-- your apache id --></username>
      <password>{encryptedpassword}</password>
    </server>
  </servers>
...
```

Defining JAVA installations

The HttpCore pom defines a couple of profiles (java-1.3, java-1.4) which configure the compiler and test cases to be run with a different version of Java from the one used to run Maven. These profiles need to know where Java 1.3 and 1.4 are installed; this is done by defining the properties JAVA_1_3_HOME and JAVA_1_4_HOME. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<settings>
...
  <properties>
    <JAVA_1_3_HOME>C:\jdk1.3.1_20</JAVA_1_3_HOME>
  </properties>
  <!-- Or, define in terms of an OS environment variable, in this case JAVA142_HOME -->
  <properties>
    <JAVA_1_4_HOME>${env.JAVA142_HOME}</JAVA_1_3_HOME>
  </properties>
...
</settings>
```

This works, however the properties will be defined for all builds, so it is best to enclose the properties in profiles so that they are only defined as necessary:

```
<?xml version="1.0" encoding="UTF-8"?>
<settings>
...
  <profiles>
    <!--
       Use profiles for the JAVA properties so they are only defined where necessary
       Note: the profile <id> entries must agree with the profiles defined in httpcore/pom.xml
      -->
    <profile>
      <id>java-1.3</id>
      <properties>
        <JAVA_1_3_HOME>C:\jdk1.3.1_20</JAVA_1_3_HOME>
      </properties>
    </profile>
    <profile>
      <id>java-1.4</id>
      <!-- Or, define in terms of an OS environment variable, in this case JAVA142_HOME -->
      <properties>
        <JAVA_1_4_HOME>${env.JAVA142_HOME}</JAVA_1_3_HOME>
      </properties>
    </profile>
   </profiles>
...
</settings>
```