Animal Welfare Assessment Grid (AWAG) Installation Guide

Contents

Prerequisites 2
Postgres database server 3
Glassfish application server 4
Apache web server 5
Authentication Options 6
Installing the application 11
Run application 12

The following information will help you to install the Animal Welfare Assessment Grid onto your organisations IT infrastructure. The following details outline an installation on a single machine; however the software can be installed across multiple machines.

Prerequisites

This guide assumes that you have the following installed:

- PostgreSQL database server: Stores the application data and/or user authentication data.
  - PostgreSQL 9.3 and 9.4 were used during the development of AWAG.
- Glassfish application server (Java EE Full Platform version): Hosts the server code, manages authentication and database access.
- Glassfish 4 and 4.1.1 were used during the development of AWAG.
- A web server: Serves the client code and acts as a reverse proxy.
  - Apache web server 2.2 was used during the development of AWAG.
- JDK 7 or above: Needed to run Glassfish/JavaEE applications.

The guide also assumes that you have downloaded the AWAG project release contents from the GitHub repository (https://github.com/PublicHealthEngland/animal-welfare-assessment-grid/releases). The guide refers to location of the download as **{github-base}**.
The contents of the release are different to the result of cloning the repository with Git 'clone' command. In addition to source code, the release also contains a zipped web application archive file (a war file). You will need the war file if you are not intending to build the AWAG from source with Maven.

{glassfish-base} refers to the root directory of the Glassfish install location.

Postgres database server

Once this section of the guide has been completed you should have:

1. A postgres installation with two databases installed.
2. A user that is able to access each of the databases

Note: the username and password will be needed in later sections of this guide.

Steps

Once Postgres has been downloaded and installed, you will need to perform the following:

1. Create a new login role named 'awag' with a password of your choice.
2. Create the following databases and make the 'awag' user that you have created is the owner of each:
    - awdatabase – holds the main database that the software uses.
    - awauth – holds the login information for users of the system if not using active directory to manage user accounts.

3. Restore the db-init.sql script to awdatabase using pgadmin or move to the bin directory of your postgres installation and run the following command:

    psql -U awag awdatabase < {github-base}/configuration/db-init.sql

4. Restore the authentication.sql script to awauth using pgadmin or move to the bin directory of your postgres installation and run the following command:

    Note: before running the command below, please change the ownership of public schema in the 'awauth' database from the 'postgres' user to the 'awag' user.

    psql -U awag awauth < { github-base }/configuration/authentication.sql

Glassfish application server

Once this section of the guide has been completed you should have:

1. Installed the database driver in glassfish.
2. Configured the database connection pools used to access the database.
3. Configured data sources used by the application to access the database.
4. Configured a choice of authentication realms used to secure the system.

Steps

The following configuration steps will help you to configure glassfish using the default domain provided, domain1.

1. Change the admin password for glassfish; move to the bin directory of your glassfish installation and run the following command and :

   asadmin change-admin-password

   - Enter the username admin.
   - Next enter the current password which should be set to nothing, so just press enter.
   - Next enter a new password for the glassfish admin console.
   - Next, retype the password to confirm.

2. Copy the postgresql-9.3-1101.jdbc41.jar driver located in {github-base}/configuration into {glassfish-base}/glassfish/domains/domain1/lib
   a. If you have used a different version of PostgreSQL database, you may need to find and use a different JDBC library to the one specified above.
3. Open {glassfish-base}/glassfish/domains/domain1/config/domain.xml, complete the xml snippet below and copy it anywhere inside the resources tag.

```
<resources>
….
<jdbc-connection-pool driver-classname="org.postgresql.Driver" ping="true" name="awDatabase" res-type="java.sql.Driver">
    <property name="password" value="{your database user password here}"></property>
    <property name="user" value="awag"></property>
    <property name="URL" value="jdbc:postgresql://localhost:5432/awdatabase"></property>
  </jdbc-connection-pool>
  <jdbc-resource pool-name="awDatabase" jndi-name="jdbc/awDatabase"></jdbc-resource>
  <jdbc-connection-pool driver-classname="org.postgresql.Driver" ping="true" name="awAuth" res-type="java.sql.Driver">
    <property name="password" value="{your database user password here]}"></property>
    <property name="user" value="awag"></property>
    <property name="URL" value="jdbc:postgresql://localhost:5432/awauth"></property>
  </jdbc-connection-pool>
  <jdbc-resource pool-name="awAuth" jndi-name="jdbc/awAuth"></jdbc-resource>
…
</resources>
```

4. Reload the configuration by restarting the glassfish domain or server.

Apache web server

Once this section of the guide has been completed you should have:

1. Client side code installed on the web server.
2. Reverse proxy set up to allow the client side code to talk to the server side code.

Steps

1. Locate the Apache2 httpd-vhosts.conf file in {apache-install-base}/conf/extra and edit it and add the following:

```
NameVirtualHost *:80
<VirtualHost *:80>
   ServerAdmin webmaster@virthost01.local
   DocumentRoot "C:/www"
   ServerName virthost01
   ErrorLog "logs/virthost01-error.log"
   CustomLog "logs/virthost01.log" common
   ProxyPass /animal-welfare-system-client/server/ http://localhost:8080/animal-welfare-system/
   ProxyPassReverse /animal-welfare-system-client/server/ http://localhost:8080/animal-welfare-system/
   ProxyPassReverseCookiePath /animal-welfare-system /animal-welfare-system-client/server/
</VirtualHost>
```

2. Copy the client side code from {github-base}/code/client/ into {apache-install-base}/www/.
3. Restart apache.

Authentication Options

Once this section of the guide has been completed you should have:

1. Glassfish configuration entry to allow for the chosen method of authentication.
2. Access to the system

Steps

There are two methods of authentication that the system uses either:

- Active directory
- JDBC authentication

Active directory

Using active directory as the authentication system means that users can login using the same credentials across multiple systems; these details can be supplied by your IT department. There are many articles online explaining how to configure glassfish to work with active directory.

1. Locate {glassfish-base}/glassfish/domains/domain1/config/domain.xml
2. Complete the xml snippet below and copy it between the security-service xml tags.

```
<security-service>
…
    <auth-realm classname="com.sun.enterprise.security.auth.realm.ldap.LDAPRealm"
name="ldapRealm">
      <property name="directory" value="ldap://{ip address of ldap server}:389"></property>
      <property name="base-dn" value="dc={base-dn  of active directory account}"></property>
      <property name="jaas-context" value="ldapRealm"></property>
      <property name="search-bind-dn" value="{name of active directory account}"></property>
      <property name="search-bind-password" value="{password of active directory
account}"></property>
      <property name="group-search-filter"
value="(&amp;(objectClass=group)(member=%d))"></property>
      <property name="search-filter"
value="(&amp;(objectClass=user)(sAMAccountName=%s))"></property>
      <property name="java.naming.referral" value="ignore"></property>
    </auth-realm>
…
</security-service>
```

3. Open the .war file located in {github-base} and ensure that the web.xml found in the WEB-INF directory contains the following:

   Note: The .war file is just a zip file so you can use 7zip's or similar to open-archive functionality to gain access to the contents.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://java.sun.com/xml/ns/javaee"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
        version="3.0">
        <display-name>aw</display-name>
        <welcome-file-list>
                <welcome-file>index.html</welcome-file>
        </welcome-file-list>
        <login-config>
                <auth-method>FORM</auth-method>
                <realm-name>ldapRealm</realm-name>
                <form-login-config>
                        <form-login-page>/login.html</form-login-page>
                        <form-error-page>/login-failed.html</form-error-page>
                </form-login-config>
        </login-config>
        <!--
        <login-config>
                <auth-method>FORM</auth-method>
                <realm-name>jdbcRealm</realm-name>
                <form-login-config>
                        <form-login-page>/login.html</form-login-page>
                        <form-error-page>/login-failed.html</form-error-page>
                </form-login-config>
        </login-config>
        -->
        <security-constraint>
                <web-resource-collection>
                        <web-resource-name>Secure Pages</web-resource-name>
                        <url-pattern>/*</url-pattern>
                </web-resource-collection>
                <auth-constraint>
                        <role-name>admin</role-name>
                </auth-constraint>
        </security-constraint>
        <security-constraint>
                <web-resource-collection>
                        <web-resource-name>Open Content</web-resource-name>
                        <url-pattern>/resources/*</url-pattern>
                </web-resource-collection>
        </security-constraint>
        <session-config>
                <session-timeout>30</session-timeout>
        </session-config>
```

```
</web-app>
```

4. Open the .war file located in {github-base} and ensure that glassfish-web.xml contains the following:

Note: The .war file is just a zip file so you can use 7zip's or similar to open-archive functionality to gain access to the contents.

```
<!DOCTYPE glassfish-web-app PUBLIC "-//GlassFish.org//DTD
GlassFish Application Server 3.1 Servlet 3.0//EN"
"http://glassfish.org/dtds/glassfish-web-app_3_0-1.dtd">
<glassfish-web-app>
        <security-role-mapping>
                <role-name>admin</role-name>
                <group-name>{active directory group name}</group-name>
        </security-role-mapping>
        <!--
        <security-role-mapping>
                <role-name>admin</role-name>
                <group-name>admin</group-name>
        </security-role-mapping>
        -->
</glassfish-web-app>
```

JDBC

In this case username and passwords will be stored in a SQL database. When a user attempts to login the application server will look up the user's credentials from its JDBC realm allowing it to check for the existence of the user and whether the password is correct. If you have been following this guide through you would have already created the authentication database in the 'postgres database server' section.

1. Open the .war located in {github-base} and ensure that the web.xml found in the WEB-INF directory contains the following:

Note: The .war file is just a zip file so you can use 7zip's or similar to open-archive functionality to gain access to the contents.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://java.sun.com/xml/ns/javaee"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
        version="3.0">
        <display-name>aw</display-name>
```

```
        <welcome-file-list>
                <welcome-file>index.html</welcome-file>
        </welcome-file-list>
<!--    <login-config>
                <auth-method>FORM</auth-method>
                <realm-name>ldapRealm</realm-name>
                <form-login-config>
                        <form-login-page>/login.html</form-login-page>
                        <form-error-page>/login-failed.html</form-error-page>
                </form-login-config>
        </login-config> -->
         <login-config>
                <auth-method>FORM</auth-method>
                <realm-name>jdbcRealm</realm-name>
                <form-login-config>
                        <form-login-page>/login.html</form-login-page>
                        <form-error-page>/login-failed.html</form-error-page>
                </form-login-config>
        </login-config>
        <security-constraint>
                <web-resource-collection>
                        <web-resource-name>Secure Pages</web-resource-name>
                        <url-pattern>/*</url-pattern>
                </web-resource-collection>
                <auth-constraint>
                        <role-name>admin</role-name>
                </auth-constraint>
        </security-constraint>
        <security-constraint>
                <web-resource-collection>
                        <web-resource-name>Open Content</web-resource-name>
                        <url-pattern>/resources/*</url-pattern>
                </web-resource-collection>
        </security-constraint>
        <session-config>
                <session-timeout>30</session-timeout>
        </session-config>
</web-app>
```

2. Open the .war file found in {github-base} and ensure that glassfish-web.xml contains the following:

   Note: The .war file is just a zip file so you can use 7zip's or similar to open-archive functionality to gain access to the contents.

```
<!DOCTYPE glassfish-web-app PUBLIC "-//GlassFish.org//DTD
```

```
GlassFish Application Server 3.1 Servlet 3.0//EN"
"http://glassfish.org/dtds/glassfish-web-app_3_0-1.dtd">
<glassfish-web-app>
<!--    <security-role-mapping>
            <role-name>admin</role-name>
            <group-name>{active directory group name}</group-name>
        </security-role-mapping> -->
        <security-role-mapping>
            <role-name>admin</role-name>
            <group-name>admin</group-name>
        </security-role-mapping>
</glassfish-web-app>
```

Installing the application

Once this section of the guide has been completed you should have:

1. Final configuration steps completed.
2. Application deployed onto glassfish.

Steps

1. Open the .war file and edit index.html. Locate the code below and replace localhost with the domain name that points to the server you are installing apache on.

   e.g. window.location.assign("http://{your.domain.here}/animal-welfare-system-client/index.html");

   Note: The .war file is just a zip file so you can use 7zip's or similar to open-archive functionality to gain access to the contents.

2. Locate global-config.js in the client code and change the 'serverUrl' JavaScript property to point to the same domain name as the previous step.

```
e.g. window.awconfig = {
   // Reverse proxy maps the two URLs below
   // serverUrl : 'http://localhost:8080/animal-welfare-system/'
   serverUrl : 'http://{your.domain.here}/animal-welfare-system-client/server/'
};
```

3. Copy the .war file into your domain auto deploy directory. e.g. {glassfish-base}/glassfish/domains/domain1/autodeploy/

Run application

Check that the installation was installed correctly by visiting the newly installed site. The URL should look similar to the following http://{your.domain.here}/animal-welfare-system-client/index.html#/main. You should be redirected to the login page. If you are unsure of what to do once you have installed the software, please visit the user guide document stored in the GitHub repository.