Get Involved

java-net Project
Request a Project
Project Help Wanted Ads
Publicize your Project
Submit Content

Get Informed

About java.net
Articles
Weblogs
News
Events
Also in Java Today
java.net Online Books
java.net Archives

Get Connected

java.net Forums
Wiki and Javapedia
People, Partners, and Jobs
Java User Groups
RSS Feeds

Search

Web and Projects: »
Online Books: »
Advanced Search

Home | Changes | Index | Search | Go

Project Wonderland v0.5: Configuring Authentication

by Jonathan Kaplan (kaplanj@dev.java.net)

DRAFT - UNDER CONSTRUCTION

Purpose

This tutorial is for system administrators who would like to enable authentication on their Project Wonderland server.

Project Wonderland supports three modes of authentication. The default is no authentication, where anyone can log in using any user id. This mode is intended only for developers. The second mode is to always require authentication. In this case, all users must have a valid account to log in to the Project Wonderland server. The last option extends the authenticated mode to allow for guest login. This means that only authenticated users can access the web administration and other sensitive areas of Project Wonderland, but other users may join as a guest. This tutorial will cover setting up the latter two authentication modes.

Project Wonderland also allows developers to add new authentication mechanisms, like LDAP, in the form of Project Wonderland modules. This tutorial will also cover how to configure these custom authenticators, using LDAP as an example.

Prerequisites

Here's what you will need to complete this tutorial:

- A running Project Wonderland server
  - From a binary download - Project Wonderland v0.5 Binary Downloads
  - Or built from source - Build Project Wonderland 0.5 From Source
- Project Wonderland v0.5, Preview2 or later - download binaries

Installing the security-session-auth module

The default no-authentication security is provided by a Project Wonderland module called "security-session-noauth". The first step in setting up security on a Project Wonderland server is to replace the security-session-noauth module with a version that supports authentication, in this case "security-session-auth". The mechanisms for doing this vary depending on if you are using a binary build of Project Wonderland or are building from source.

...in a binary download

Download a copy of the security-session-auth.jar module file from the Project Wonderland module warehouse. Note that the real module warehouse is currently under construction, so you should download the module from the folder in the temporary warehouse that is appropriate to your version of Project Wonderland. For a nightly build, this is the "0.5-dev" folder. For the preview 2 release, this is the "preview2" folder. The version in the preview1 folder is out-of-date, and may not work with this tutorial.

Now, use the module management UI to remove the module named "security-session-noauth" and replace it by uploading the security-session-auth.jar module file that you downloaded. For more information on managing modules, see managing modules. When you are done, your module list should look something like this (Click on the image to see a full-screen version):

<< IMAGE >>

As long as you continue to run the same binary version of Project Wonderland, everything should work correctly. If you update to a newer nightly build or release binary, you will need to go in and remove the security-session-noauth.jar again.

That's it. You can now skip directly to Configuring Authentication.

...in a source build

In a source build, we will disable building the security-session-noauth module and add the security-session-auth module to the

default list of modules to build. We will do this by editing the build.xml files in the appropriate directories. This assumes you have a directory structure with "Project Wonderland" and "Project Wonderland-modules" checked out as siblings:

```
<top-dir>
<top-dir>/Project Wonderland
<top-dir>/Project Wonderland-modules
<top-dir>/Project Wonderland-modules/stable
<top-dir>/Project Wonderland-modules/unstable
```

The first step is to disable security-session-noauth from building. We do this in <top-dir>/Project Wonderland/modules/tools/build.xml, by commenting out the line as shown below:

```
    <!-- all projects -->
    <filelist dir="${current.dir}" id="all.projects">
        <file name="darkstar/build.xml"/>
        <file name="security-group/build.xml"/>
        <file name="security-session-common/build.xml"/>
        <!--file name="security-session-noauth/build.xml"/-->
        <file name="presence-manager/build.xml"/>
```

The next step is to add security-session-auth to the default packaged jars. We do this in <top-dir>/Project Wonderland-modules/stable/build.xml, by adding the security-session-auth line below:

```
    <!--
         copy only the selected modules to the dist directory.  This is the
         default action, unless the modules.include.all property is set
    -->
    <target name="-modules-stable-dist-copy-selected" unless="modules.include.all">
        <copy todir="dist">
            <filelist dir=".">
                <file name="orientationworld/dist/orientationworld.jar"/>
                <file name="pdfviewer/dist/pdfviewer.jar"/>
                <file name="telepointer/dist/telepointer.jar"/>
                <file name="whiteboard/dist/whiteboard.jar"/>
                <file name="image-viewer/dist/imageviewer.jar"/>
                <file name="audiorecorder-module/dist/audiorecorder.jar"/>
                <file name="stickynote/dist/stickynote.jar"/>
                <file name="security-session-auth/dist/security-session-auth.jar"/>
```

Finally, before we rebuild we need to make sure there are no cached copies of security-session-noauth.jar hanging around. From the top directory:

```
% rm Project Wonderland/modules/dist/security-session-noauth.jar
% cd Project Wonderland/modules/tools/security-session-noauth
% ant clean
```

Configuring authentication

Now that the modules are set up correctly, restart the Project Wonderland server ("java -jar Project Wonderland.jar" in binary or "ant run-server" in a source build). From this point forward, the instructions should be basically the same for binary distributions and source builds.

Connect to the web UI, and select "Server Admin". If all goes well, you should be prompted to login, as shown below:

<< IMAGE >>

Login using the default username, "admin", and the default password, also "admin". Since this isn't very secure, our first task is to change the password. Once you have logged in, select "User Manager" from the list of pages on the left. This will bring up a list of default users, as shown below (Click on the image to see a full-screen version):

<< IMAGE >>

These are the default user accounts used by the services in the system like the Darkstar server, the web server and the shared application server. Select the "edit" link next to each user to bring up an editor for that user, as shown below:

<< IMAGE >>

Change the password, and then click "update user". While you are here, this would also be a good time to add some accounts for the users who will log in. Depending on your login setup, you may choose to define accounts in the User Manager, or using separate mechanisms like LDAP (more on that later). Below, I've created a user account for myself:

<< IMAGE >>

Configure services

Next up, we have to tell the various services like the Darkstar server and the Shared App Server about these new passwords we just set. For each password that you set above, create a password file somewhere in the filesystem of your server. This file should consist of a single line with the password you used. The files can be put anywhere, but since they contain an actual password,

make sure they are protected for read access. I tend to put the password files in the ~/.wonderland-server directory, with all the other server data, and protect them with normal file permissions. On my Mac, this looks like:

```
% cd ~/.wonderland-server/0.5-dev
% echo "supersecretpassword" > wonderland.password
% chmod 400 wonderland.password
% pwd
/Users/jkaplan/.wonderland-server/0.5-dev
```

Now my password file is in /Users/jkaplan/.wonderland-server/0.5-dev/wonderland.password. I only created a single password file, because I use the same password for all the various services. Depending on your security needs, you can use different passwords and password files for each service if you prefer.

Head back to the web UI and click on the "manage servers" link. For each service, we'll need to set a password by clicking the "edit" link next to that service. Each service uses a different property to set the password file. Set the appropriate property for the service (see table below) to the full path to the password file you created above. For the Darkstar server, it should look something like (Click on the image to see a full-screen version):

<< IMAGE >>

Right now, you only need to configure passwords for two services:

Component	Property
Darkstar Server	sgs.password.file
Shared App Server	sas.password.file

The final step is to configure the admin password in the web server itself. You do this by editing the my.run.properties file (see here for more information on setting properties). Add the following property:

```
wonderland.webserver.password.file=/Users/jkaplan/.wonderland-server/0.5-dev/wonderland.
```

Make sure that this points to the password for the "webserver" user you configured earlier.

Restart with security enabled

Now restart the Wonderland web server one more time, and you should be up and running with security. As before, when you login in to the server using the "Server Admin" link, you should be prompted for a password. Use the password you configured for the admin user above. When you log in, you should now see that all the services are properly running.

If everything has worked as expected, when users log in, they should be prompted for a password:

<< IMAGE >>

User can log in using the accounts set up by the administrator in the user manager. Read on for more advanced configuration, including setting up different authentication mechanisms.

(Optional) Enable guest login

Now that security is turned on, the sensitive areas of Wonderland like the web administration UI are protected, but every user requires a password to connect. With security enabled, you can turn on guest login to allow users to log in without an account. These logins can be made with any user name, as long as it is not already in use by another authentication mechanism (so guests cannot pretend to be users defined in the user manager).

To enable guest login, simple add the following property to the my.run.properties file:

```
wonderland.security.allow.guest.login=true
```

Restart the Wonderland web server, and guest login will be enabled. When users go to log in, they will now have the choice to use an authenticated or unauthenticated login, as shown below:

<< IMAGE >>

(Optional) Add different authentication mechanisms

By default, when you add security to Wonderland, it looks for user accounts from the list of accounts you created in the user manager web UI. For many setups, it is desirable to use other mechanisms to authenticate users, such as an LDAP directory. To support this, the Wonderland security system allows administrators to add new user plugins to configure security.

A user plugin is a Wonderland module that implements a particular interface for accessing remote authentication servers (see Developing with the Wonderland 0.5 security system for details on how to build user plugins). The user plugin verifies a user's credentials and returns back details about that user, like their name and email address.

The Wonderland web server maintains a list of user plugins. Whenever a user authenticates, each plugin in the list is queried to see if it supports the user with the given credentials. The plugin must give one of three responses:

1. MATCH - the user ID and credentials are a match with this plugin. Accept the login.

2. NO MATCH - the user ID is known to this plugin, but the credentials don't match. Reject the login.

3. UNKNOWN - the user ID is not known to this plugin. Continue on to the next plugin.

Wonderland ships with two plugins by default. The DBUserPlugin ? uses the database created by the Wonderland user manager web UI to authenticate users. The GuestUserPlugin ? is used for guest logins, and accepts all users that are passed in to it. Additional plugins can be found in the Wonderland module warehouse. For example, the LDAPUserPlugin ?.jar module provides a user plugin that queries an LDAP directory for authentication information.

A typical setup would be to use the DBUserPlugin ?, followed by the LDAPUserPlugin ?, followed by the GuestUserPlugin ?. Using this setup, users specified in the Wonderland database (like the admin, webserver and darkstar users needed by the system) would be evaluated first. If the user isn't found, their credentials would be checked against the LDAP database. Finally, if the user id doesn't exist in LDAP, the user would be treated as a guest. This setup lets users with LDAP credentials connect securely, but also allows less secure connections for guests. Using group permissions, it would further be possible to make sure that only LDAP users had access to modify the world, while guests could only view the world. Note that if the GuestUserPlugin ? is used, it should always be last on the list (because it accepts all logins), and the "wonderland.security.allow.guest.login" property described above should be set as well.

To set up different user plugins, first install the appropriate plugin module. Then create a file in the filesystem that describes the configuration of user plugins. Here is a plugin configuration file for the setup above. Like the password files above, you can put this file anywhere, but I tend to put it in the ~/.wonderland-server directory.

<user-plugin-list> <user-plugin> org.jdesktop.wonderland.modules.securitysession.auth.weblib.DBUserPluginImpl </user-plugin> <user-plugin> org.jdesktop.wonderland.modules.ldaplogin.weblib.LDAPUserPlugin ldaps://directory.java.net ou=People,dc=wonderland,dc=org </user-plugin> <user-plugin> org.jdesktop.wonderland.modules.securitysession.auth.weblib.GuestUserPluginImpl </user-plugin> </user-plugin-list>

As you can see, each plugin is listed with a user-plugin entry. The required value is the class name of the user plugin, specified by the class tag in the configuration file. The LDAP plugin also has some configuration, in the form of key / value pairs. You can see these values set in the optional properties section for the entry. See the documentation for a specific provider for more information on configuration for that particular plugin.

Once you have configured your plugin file, you just need to tell the Wonderland web server to use it. You do this by adding an entry to the my.run.properties file:

```
AuthSessionManagerImpl.UserPluginConfig=/Users/jkaplan/.wonderland-server/0.5-dev/wonder
```

Now when you start Wonderland, users will be able to authenticate with their LDAP credentials.

(Developers) Enable ant deploy

Developers might notice that the "ant deploy" task no longer works, because security is required to deploy modules. You can always use the web UI to deploy modules manually, but it is often more convenient to re-enable ant deploy. To do this, edit the my.build.properties file in your wonderland/ source directory. If the file doesn't exist, you can create it -- it is the equivalent of my.run.properties for build-time settings, and goes in the same directory in binary builds. Add the following lines to the file:

```
wonderland.web.username=admin
wonderland.web.password=supersecretpassword
```

The password is stored directly in this file, so make sure to protect the file if necessary by changing its permissions.