IHTSDO Identity Management Service

Overview

Documentation on integration with the IHTSDO Identity Management Service.

Details

This tool has a configurable security architecture that allows for a variety of possible security implementations to be used. For IHTSDO deployments of the tool, we make use of a handler (and login page override) that interacts with the identity management service.

The security layer itself assigns auth tokens which get passed back-and-forth between client and server. This handler verifies that the user is indeed logged in (or redirects them to the "guest" view of the tool). If logged in, an auth token from the application is requested that is good until the timeout period is reached.

There are several important parts to how this works:

Setting Configuration

In the deployment config.properties file, the security handler must be set to an IMS handler that is properly configured. For example:

```
security.timeout=7200000
security.guest.disabled=false
security.handler=IMS
security.handler.IMS.class=org.ihtsdo.otf.refset.jpa.services.handlers.I
msSecurityServiceHandler
security.handler.IMS.url=https://ims.ihtsdotools.org
```

The ImsSecurityServiceHandler is used with a IMS URL of ims.ihtsdotools.org

Nginx setup

In order for this all to work the system must be deployed on an ihtsdotools.org server which has a local "ims-api" link that redirects properly to IMS. For example,

```
# in the nginx configuration file for refset deployment

  location /ims-api {
      proxy_pass https://ims.ihtsdotools.org/api;
  }
```

This allows the application to check this URL to determine if the user is, in fact logged in.

Custom loginController.js

The normal login controller handles a username/password form. For IMS security, an override of the login controller (in the config/prod project) is used to customize the functionality. Instead, it first checks whether the user is logged in by probing the /ims-api link (shown above).

- If the user is NOT logged in, they are redirected to the specified "security.handler.IMS.url".
- If the user IS logged in, the username and account info from the /ims-api call are sent as username/password to the "authenticate"

call, which is handled by the ImsSecurityServiceHandler in the background, which assigns a local application auth token to the user which is used for subsequent calls.

Supporting DEV Environment

NOTE: this is not a normal thing to need to do - default build with username/password security is recommended for DEV environments.

In a local dev environment, IMS security can also be used. In this case, the user must set the deployment URL (which is presumed to be running on http://localhost:8080/refset-rest) to instead be https://local.ihtsdotools.org/refset-rest (which redirects back to localhost:8081).

The local dev environment then needs to have a configuration setup something like this (to support the localhost:8081 redirection back to localhost:8080 and the /ims-api link)

```
http {
    include    mime.types;
    server {
        listen      8081;
        server_name localhost;

        location / {
            proxy_pass http://127.0.0.1:8080;
        }

   # may need to be uat-ims for uat-authoring
        location /ims-api {
            proxy_pass https://ims.ihtsdotools.org/api;
        }

    }
}
```

In order for this to work, nginx must be running on the dev machine.

Finally, to truly use IMS security in a local dev environment, the project will need to be built locally with "-Dconfig.artifactId=refset-config-prod" passed to Maven so the overridden login controller can be deployed.

Alternatively, if IMS is needed simply to support pass-through of auth tokens (e.g. for testing Snowowl terminology handler), then the default login setup can be used (with normal username/password security) and as long as the user is actually logged into IMS and the application is being run on https://local.ihtsdotools.org, then the tokens should be successfully passed through to the handler.