APS Administration Web

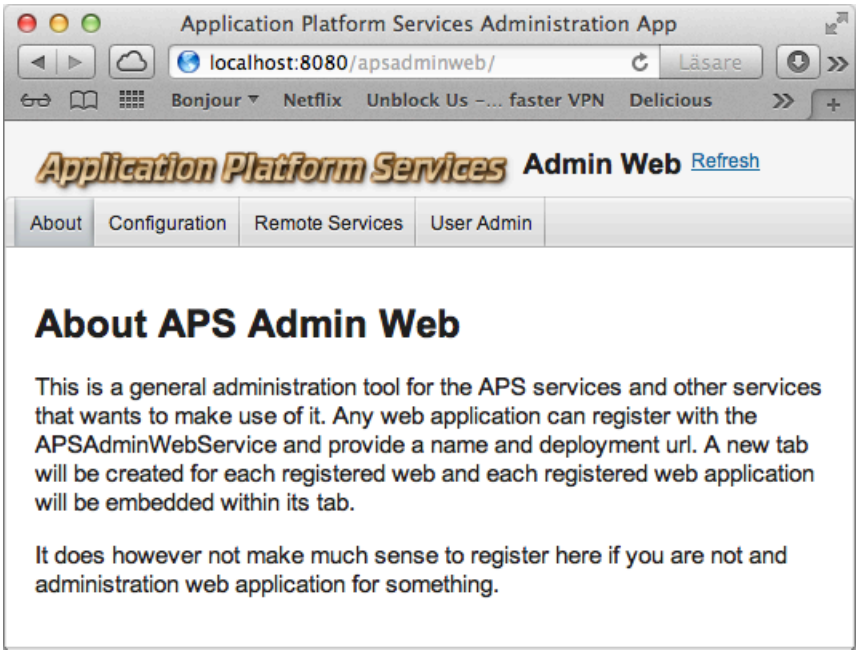
User Guide

Version: 0.9.0

Author: Tommy Svensson

Table of Contents

1 APSAdminWeb



This is a web app for administration of APS. It is really only a shell for different administraion webs. It relys on the *aps-admin-web-service-provider* bundle which publishes the *APSAdminWebService*. Other bundles providing administration web apps register themselves with this service and for each registration APSAdminWeb creates a tab in its gui. Se *APIs* further down for the APSAdminService API. Clicking on ”Refresh” will make APSAdminWeb reload the admin webs registered in *APSAdminWebService*.

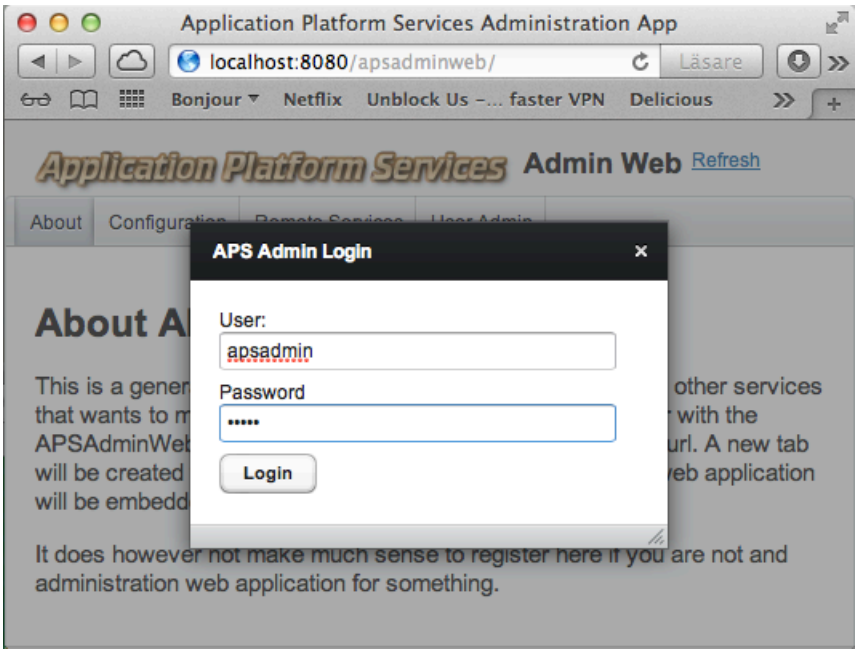
The APSAdminWeb is accessed at **http://host:port/apsadminweb**. What you see there depends on what other admin webs are deployed. Anybody can make an admin web and register it with the *APSAdminWebService*. The admin webs delivered with APS are mainly done using Vaadin. This is in no way a requirement for an admin web. An admin web app can be made in any way what so ever. A side effect of this is that different tabs might have different look and feel. But I offer that for flexibility.

The following APS bundles provides a tab in APSAdminWeb:

- *aps-config-admin-web.war* - Allows advanced configuration of bundles/services using APSConfigService.
- *aps-user-admin-web-war* - Administration of users and groups for APSSimpleUserService.
- *aps-ext-protocol-http-transport-provider.war* - Provides a web gui with help for setting up and calling services remotely, and also shows all available services and allows calling them from the web gui for testing/debugging purposes.

1.1 Authentication

The APSAdminWeb requires a login to be accessed. A userid and a password will be asked for. The entered information will be validated by the APSAuthService. The *aps-simple-user-service-auth-service-provider.jar* bundle provides an implementation of this service that uses the *APSSimpleUserService* service. The APSAuthService is however simple enough to implement yourself to provide login to whatever you want/need.



1.2 Making an admin web participating in the APSAdminWeb login.

There is an APSSessionService that was made just for handling this. It is not a HTTP session, just a service hanling sessions. It is provided by the *aps-session-service-provider.jar* bundle. When a session is created you get a session id (an UUID) that needs to be passed along to the other admin webs through a cookie. *APSWebTools* (*aps-web-tools.jar* (not a bundle!)) provides the APSAdminWebLoginHandler class implementing the LoginHandler interface and handles all this for you.

You need to provide it with a BundleContext on creation since it will be calling both the *APSAuthService* and *APSSessionService*:

```
this.loginHandler = new APSAdminWebLoginHandler(bundleContext);
```

Then to validate that there is a valid login do:

```
this.loginHandler.setSessionIdFromRequestCookie(request);
if (this.loginHandler.hasValidLogin()) {
    ...
}
else {
    ...
}
```

1.3 APSAdminWebService APIs

public *interface* **APSAdminWebService** [se.natusoft.osgi.aps.apsadminweb.service] {

This service registers other specific administration web applications to make them available under a common administration gui.

public void registerAdminWeb(AdminWebReg adminWebReg) throws IllegalArgumentException

Registers an admin web application.

Parameters

adminWebReg - *Registration information for the admin web.*

Throws

IllegalArgumentException - *if the admin web has already been registered or if it is using the*

public void unregisterAdminWeb(AdminWebReg adminWebReg)

Unregisters a previously registered admin web. This is failsafe. If it has not been registered nothing happens.

Parameters

adminWebReg - *Registration information for the admin web. Use the same as registered with.*

public List<AdminWebReg> getRegisteredAdminWebs()

Returns

All currently registered admin webs.

}

---

public *class* **AdminWebReg** [se.natusoft.osgi.aps.apsadminweb.service.model] {

This model holds information about a registered admin web application.

public AdminWebReg(String name, String version, String description, String url)

Creates a new AdminWebReg instance.

Parameters

name - *A (short) name of the admin web.*

version - *The version of the admin web.*

description - *A longer description of the admin web.*

url - *The deployment url of the admin web.*

public String getName()

Returns

The (short) name of the admin web.

public String getVersion()

Returns

The version of the admin web.

public String getDescription()

Returns

The description of the admin web.

public String getUrl()

Returns

The deployment url of the admin web.

}
