APS Web Tools

User Guide

Version: 0.9.0

Author: Tommy Svensson

Table of Contents

1 APSWebTools

This is not an OSGi bundle! This is a plain jar containing utilities for web applications. Specifically APS administration web applications. This jar has to be included in each web application that wants to use it.

Among other things it provides support for being part of the APS administration web login (APSAdminWebLoginHandler). Since the APS administration web is built using Vaadin it has Vaadin support classes. APSVaadinOSGiApplication is a base class used by all APS administration webs.

1.1 APIs

The javadoc for these can be found at http://apidoc.natusoft.se/APSWebTools.

public *class* **APSAdminWebLoginHandler** extends APSLoginHandler implements APSLoginHandler.HandlerInfo [se.natusoft.osgi.aps.tools.web] {

This is a login handler to use by any admin web registering with the APSAdminWeb to validate that there is a valid login available.

public APSAdminWebLoginHandler(BundleContext context)

Creates a new APSAdminWebLoginHandler.

Parameters

context *- The bundle context.*

public void setSessionIdFromRequestCookie(HttpServletRequest request)

Sets the session id from a cookie in the specified request.

Parameters

request *- The request to get the session id cookie from.*

public void saveSessionIdOnResponse(HttpServletResponse response)

Saves the current session id on the specified response.

Parameters

response *- The response to save the session id cookie on.*

}

public *class* **APSLoginHandler** implements LoginHandler [se.natusoft.osgi.aps.tools.web] {

This class validates if there is a valid logged in user and also provides a simple login if no valid logged in user exists.

This utility makes use of APSSimpleUserService to login auth and APSSessionService for session handling. Trackers for these services are created internally which requires the shutdown() method to be called when no longer used to cleanup.

The bundle needs to import the following packages for this class to work:

```
<code>
   se.natusoft.osgi.aps.api.auth.user;version="[0.9,2)",
   se.natusoft.osgi.aps.api.auth.user.model;version="[0.9,2)",
   se.natusoft.osgi.aps.api.misc.session;version="[0.9,2)"
</code>
```

public APSLoginHandler(BundleContext context, HandlerInfo handlerInfo)

Creates a new VaadinLoginDialogHandler.

Parameters

context - The bundles BundleContext.

protected void setHandlerInfo(HandlerInfo handlerInfo)

Sets the handler info when not provided in constructor.

Parameters

handlerInfo - The handler info to set.

public void shutdown()

Since this class internally creates and starts service trackers this method needs to be called on shutdown to cleanup!

public String getLoggedInUser()

This returns the currently logged in user or null if none are logged in.

public boolean hasValidLogin()

Returns true if this handler sits on a valid login.

public boolean login(String userId, String pw)

Logs in with a userid and a password.

Returns

true if successfully logged in, false otherwise.

Parameters

userId *- The id of the user to login.*

pw *- The password of the user to login.*

public boolean login(String userId, String pw, String requiredRole)

Logs in with a userid and a password.

This method does not use or modify any internal state of this object! It only uses the APSUserService that this object sits on. This allows code sitting on an instance of this class to use this method for validating a user without having to setup its own service tracker for the APSUserService when this object is already available due to the code also being an APSAdminWeb member. It is basically a convenience.

Returns

a valid User object on success or null on failure.

Parameters

userId *- The id of the user to login.*

pw *- The password of the user to login.*

requiredRole *- If non null the user is required to have this role for a successful login. If it doesn't null will*

public *static interface* **HandlerInfo** [se.natusoft.osgi.aps.tools.web] {

Config values for the login handler.

String getSessionId()

Returns

An id to an APSSessionService session.

void setSessionId(String sessionId)

Sets a new session id.

Parameters

sessionId *- The session id to set.*

String getUserSessionName()

Returns

The name of the session data containing the logged in user if any.

String getRequiredRole()

Returns

The required role of the user for it to be considered logged in.

}

---

public *class* **ClientContext** [se.natusoft.osgi.aps.tools.web] {

A context to pass to client code for use in calling services.

public ClientContext(UserMessager userMessager, OSGiBundleContextProvider bundleContextProvider)

Creates a new ClientContext instance.

Parameters

userMessager *- Used to send messages to the user.*

bundleContextProvider *- Provides the OSGi BundleContext.*

public BundleContext getBundleContext()

Returns the OSGi BundeContext.

public UserMessager getMessager()

Use for producing messages to the user.

public <Service> void addService(Class<Service> serviceInterface, Service serviceImpl)

Adds a service to the context.

Parameters

serviceInterface *- The interface of the service to add.*

serviceImpl *- The implementation to the service.*

public <Service> Service getService(Class<Service> serviceClass)

Returns the service of the specified service interface.

Returns

The service.

Parameters

<Service> *- The service type.*

serviceClass *- The service type class.*

}

---

public *class* **CookieTool** [se.natusoft.osgi.aps.tools.web] {

Provides simple static cookie tools.

public static String getCookie(Cookie[] cookies, String name)

Returns the cookie having the specified name or null if none is found.

Parameters

cookies *- The complete set of cookies from the request.*

public static void setCookie(HttpServletResponse resp, String name, String value, int maxAge, String path)

Sets a cookie on the specified response.

Parameters

resp *- The servlet response to set the cookie on.*

name *- The name of the cookie.*

value *- The value of the cookie.*

maxAge *- The max age of the cookie.*

public void deleteCookie(String name, HttpServletResponse resp)

Removes a cookie.

Parameters

name *- The name of the cookie to remove.*

resp *- The servlet response to remove the cookie on.*

}

---

public *interface* **LoginHandler** [se.natusoft.osgi.aps.tools.web] {

This is a simple API for doing a login.

public boolean hasValidLogin()

Returns true if this handler sits on a valid login.

boolean login(String userId, String pw)

Logs in with a userid and a password.

Returns

true if successfully logged in, false otherwise.

Parameters

userId *- The id of the user to login.*

pw *- The password of the user to login.*

public void shutdown()

If the handler creates service trackers or other things that needs to be shutdown when no longer used this methods needs to be called when the handles is no longer needed.

}

---

public *interface* **OSGiBundleContextProvider** [se.natusoft.osgi.aps.tools.web] {

Gives access to the OSGi BundleContext.

public BundleContext getBundleContext()

Returns the context for this deployed bundle.

}

---

public *interface* **UserMessager** [se.natusoft.osgi.aps.tools.web] {

Handles presenting user with messages.

Different GUI choices needs different implementations of this. The basic idea behind this is to make message handling less dependent on GUI. Unit tests can also supply own implementation of this.

public void error(String caption, String message)

Shows an error message on the window.

Parameters

caption *- The message caption.*

message *- The message.*

public void warning(String caption, String message)

Shows a warning message on the window.

Parameters

caption *- The message caption.*

message *- The message.*

public void info(String caption, String message)

Shows an info message on the window.

Parameters

caption *- The message caption.*

message *- The message.*

public void tray(String caption, String message)

Shows a tray message on the window.

Parameters

caption *- The message caption.*

message - *The message.*

}

---

public *class* **APSSessionListener** implements HttpSessionListener [se.natusoft.osgi.aps.tools.web.vaadin] {

This is registered in web.xml and installs itself in the session on session create using its class name as key. Web applications can then get this from the session and register themselves as "destroyed" listeners on this to be called when sessionDestroyed() is called on this.

The main reason for this is to be able to shutdown and release any service trackers that were started on session startup.

public void addDestroyedListener(APSSessionDestroyedListener destroyedListener)

Adds a session destroyed listener to forward session destroyed events to.

Parameters

destroyedListener

public *static interface* **APSSessionDestroyedListener** [se.natusoft.osgi.aps.tools.web.vaadin] {

This should be implemented by class wanting to receive the sessionDestroyed event.

public void sessionDestroyed()

Gets called when the session is destroyed.

}

---

public *class* **APSTheme** [se.natusoft.osgi.aps.tools.web.vaadin] {

The only purpose of this class is to define the vaadin theme used.

}

---

public *abstract class* **APSVaadinOSGiApplication** extends com.vaadin.Application implements OSGiBundleContextProvider, APSSessionListener.APSSessionDestroyedListener [se.natusoft.osgi.aps.tools.web.vaadin] {

APS base class for Vaadin application providing OSGi support.

This does the following: <ul <li Looks up the bundle context in the servlet context and informs the user that it

must be deployed in an OSGi server to function if not found. li <li Creates a ClientContext containing the BundleContext, but can also be used to store services in. li <li Calls overridable initServices(ClientContext), initGUI() in that order to setup. li <li Registers a session listener and calls overridable cleanupServices() when the session dies. li <li provides getters for both the BundleContext and the ClientContext. li ul

public void init()

Initializes the vaadin application.

protected void initGUI() } / * Initializes the service setup. * * clientContext The client context for accessing services. */ protected void initServices(ClientContext clientContext) } /** * Called when the session is about to die to cleanup anything setup in initServices(). * <p/> * This method should be overriden by subclasses who need cleanup. * * clientContext The client cntext for accessing services. */ protected void cleanupServices(ClientContext clientContext) } /** * Intercepts setMainWindow() and supplies it to the VaadinUserMessager created in init() and used to * display user messages. * * mainWindow */ public void setMainWindow(Window mainWindow)**

Initializes the gui part of the applicaiton.

public ClientContext getClientContext()

Returns

The client context.

}

---

public *class* **HorizontalLine** extends Label [se.natusoft.osgi.aps.tools.web.vaadin.components] {

This is a component that draws a horizontal line.

public HorizontalLine()

Creates a new HorizontalLine.

}

---

public *class* **HTMLFileLabel** extends Label [se.natusoft.osgi.aps.tools.web.vaadin.components] {

This is a Label that takes a classpath relative path to an html file and loads it as label content. Please note that it required XHTML!

Any comment blocks in the html are skipped when loading.

public HTMLFileLabel(String htmlFilePath, String themeName, ClassLoader classLoader)

Creates a new HTMLFileLabel.

Parameters

htmlFilePath *- The label content html file.*

themeName *- The name of the theme used. Any {vaadin-theme} in loaded html will be*

classLoader *- The class loader to use for finding the html file path.*

}

---

public *interface* **MenuBuilder<MenuItemRepresentative>**
[se.natusoft.osgi.aps.tools.web.vaadin.components.menutree.builderapi] {

This must be implemented by a provider of menu entries.

public void buildMenuEntries(HierarchicalModel<MenuItemData<MenuItemRepresentative>> menuModel)

This should add menu entries to the received menu model.

Parameters

menuModel *- The model to add menu entries to.*

}

---

public *class* **MenuItemData<ItemRepresentative>**
[se.natusoft.osgi.aps.tools.web.vaadin.components.menutree.builderapi] {

Holds item related information that is associated with a model item by its item id.

public MenuItemData() } // // Methods // / This represents one specific configuration. */ public ItemRepresentative getItemRepresentative()**

Creates a new MenuTreeItemData instance.

public String getToolTipText()

The tooltip text for the item.

public Action[] getActions()

The actions for the item.

public ComponentHandler getSelectComponentHandler()

The component hander for when this item is selected.

public Map<Action, MenuActionProvider> getActionComponentHandlers()

The menu action handlers per action.

}

---

public *interface* **ComponentHandler** extends MenuActionProvider [se.natusoft.osgi.aps.tools.web.vaadin.components.menutree.handlerapi] {

Implementations of this combines possibly a singleton instance of a GUI component with item specific data and then returns the component.

public AbstractComponent getComponent()

Returns

The component that should handle the item.

}

---

public *interface* **MenuActionExecutor** extends MenuActionProvider [se.natusoft.osgi.aps.tools.web.vaadin.components.menutree.handlerapi] {

This is an alternative to ComponentHandler for when there is not need to display a component, just perform some action.

public void executeMenuAction()

Executes the menu action.

}

}

public *class* **MenuTree** extends Tree implements Action.Handler, ItemDescriptionGenerator, Refreshable [se.natusoft.osgi.aps.tools.web.vaadin.components.menutree] {

This is a semi advanced menu tree component.

You add 'MenuBuilder's to provide the menu contents. The builders also provide actions for when a menu entry is accessed.

public static final Action[] NO_ACTIONS = new Action[0]

Indicates no actions.

public MenuTree()

Creates a new MenuTree component.

public void addMenuBuilder(MenuBuilder menuBuilder)

Adds a MenuBuilder to the menu tree.

Parameters

menuBuilder - *The menu builder to add.*

public void refresh()

Reloads the contents of the menu. This is never done automatically!!

public MenuItemData getItemData(IntID itemId)

Returns a data object associated with a menu item by its item id.

Returns

The associated data or null.

Parameters

itemId - *The item id of the item whose associated data to get.*

public void setActionHandler(MenuActionHandler actionHandler)

Sets the action handler for forward actions to.

Parameters

actionHandler - *The action handler to set.*

public void expandHierarchicalModel()

Expands all nodes in the hierarchical model.

public *static interface* **MenuActionHandler** [se.natusoft.osgi.aps.tools.web.vaadin.components.menutree] {

This provides half of the Action.Handler API since we provide the first part our self.

public void handleAction(Action action, Object sender, Object target)

Handles an action for the given target. The handler method may just discard the action if it's not suitable.

Parameters

action *- the action to be handled.*

sender *- the sender of the action. This is most often the action*

target *- the target of the action. For item containers this is the*

}

---

public *class* **SidesAndCenterLayout** extends HorizontalLayout
[se.natusoft.osgi.aps.tools.web.vaadin.components] {

This is a layout that only accepts 5 components at 5 positions: left, top, center, bottom, right.

The layout looks like this:

You set the different components with "setLeft(...), setTop(...), ...". All positions are optional, and you don't have to set any, but that would be pointless.

When you have set all your components the doLayout() method must be called, and it cannot be called before that. If you fail to call doLayout() there will be an exception when rendering.

public SidesAndCenterLayout()

Creates a new SidesAndCenterLayout.

public void setLeft(Component left)

Sets the component for the left side.

Parameters

left - *The component to set.*

public Component getLeft()

Returns

The left component or null if none.

public void setTop(Component top)

Sets the component for the top.

Parameters

top - *The component to set.*

public Component getTop()

Returns

The top component or null if none.

public void setCenter(Component center)

Sets the component for the center.

Parameters

center - *The component to set.*

public Component getCenter()

Returns

The center component or null if none.

public void setBottom(Component bottom)

Sets the component for the bottom.

Parameters

bottom *- The component to set.*

public Component getBottom()

Returns

The bottom component or null if none.

public void setRight(Component right)

Sets the component for the right side.

Parameters

right *- The component to set.*

public Component getRight()

Returns

The right component or null if none.

public void doLayout()

Does the actual layout adding the provided component to the layout. This must be called after the components have been set.

It starts by removing all components so if this has already been setup and this method called, it can be called again with new components added or components replaced with other components.

}

---

public *class* **HierarchicalModel<Data>** [se.natusoft.osgi.aps.tools.web.vaadin.models] {

This is a wrapping of Vaadins HierarchicalContainer.

The reason for this is that I want to be able to associate a set of specific data with each item. For a Tree you can use item properties, but that is more of a side-effect than intentional usage. It will work less well with a Table. So this wrapper will build a HierarchicalContainer, generate item ids, which will also be used to associate a data

object with each item in a separate map. Once the HierarchicalModel is built the HierarchicalContainer can be gotten from it and the data object for any item can be looked up by its id.

This does currently not support default values!

public HierarchicalModel(ID idProvider, String... captionProperties)

Creates a new HierarchicalModel.

Parameters

idProvider *- An ID implementation providing IDs.*

captionProperties *- container captions.*

public HierarchicalModel(ID idProvider)

Creates a new HierarchicalModel.

Parameters

idProvider *- An ID implementation providing IDs.*

public static String getDefaultCaption()

If the no arg constructor is used the caption property is made up, and this will return it.

Returns

The made up caption property.

public ID addItem(ID parent, Data data, String... captions)

Adds an item to the model.

Returns

The item id of the added item.

Parameters

parent *- If non null this should be the item id of the parent of the item added.*

data *- The data to associate with the item.*

captions *- Caption values for the caption properties.*

public ID addItem(Data data, String... captions)

Adds an item to the model.

Returns

The item id of the added item.

Parameters

data - *The data to associate with the item.*

captions - *Caption values for the caption properties.*

public Data getData(ID itemId)

Returns the associated data for an item.

Returns

The associated data.

Parameters

itemId - *The id of the item to get associated data for.*

public ID getCurrentItemId()

Returns

The current item id, which is also always the last/highest item id.

public HierarchicalContainer getHierarchicalContainer()

Returns the HierarchicalContainer that we have built so far.

Please note that it is the internal instance that is returned, not a copy of it! Thereby any changes made after the call to this method will still affect the returned object!

}

---

public *interface* **Refreshable** [se.natusoft.osgi.aps.tools.web.vaadin.tools] {

This is a way to provide components that needs to be refreshed to other components without creating dependencies to the whole component.

public void refresh()

Refreshes its content.

}

---

public *class* **Refreshables** implements Iterable<Refreshable> [se.natusoft.osgi.aps.tools.web.vaadin.tools] {

Manages a set of Refreshable objects.

public Refreshables() } // // Methods // / * Adds a refreshable to the refreshables. * * refreshable The refreshable to add. */ public void addRefreshable(Refreshable refreshable)**

Creates a new Refreshables instance.

public void refresh()

Calls on the managed refreshables to do refresh.

}

---

public *interface* **RefreshableSupport** [se.natusoft.osgi.aps.tools.web.vaadin.tools] {

This can be implemented by classes that supports adding Refreshables.

public void addRefreshable(Refreshable refreshable)

Adds a refreshable to be passed to editors on menu entry selection.

Parameters

refreshable *- The refreshable to add.*

}

---

public *class* **VaadinLoginDialogHandler** [se.natusoft.osgi.aps.tools.web.vaadin] {

This is a Vaadin based login dialog.

public VaadinLoginDialogHandler(Window appWindow, LoginHandler loginHandler)

Creates a new VaadinLoginDialogHandler.

Parameters

appWindow - *The Vaadin application window to add the popup login dialog to.*

loginHandler - *A handler for doing the login from the login dialog input.*

public void setLoginDialogTitle(String loginDialogTitle)

Sets the title of the login dialog window.

Parameters

loginDialogTitle - *The title to set.*

public void doLoginDialog()

This will popup the login dialog if it is not already showing. Any previously entered user and password are ofcourse cleared!

}

---

public *class* **VaadinUserMessager** implements UserMessager [se.natusoft.osgi.aps.tools.web.vaadin] {

Implementation of UserMessager for Vaadin applications using a Vaadin Window to do showNotification(...) on.

public VaadinUserMessager()

Creates a new VaadinUserMessager instance.

public void setMessageWindow(Window messageWindow)

Sets the message window.

Parameters

messageWindow - *The message window to set.*

public void error(String caption, String message)

Shows an error message on the window.

Parameters

caption - *The message caption.*

message - *The message.*

public void warning(String caption, String message)

Shows a warning message on the window.

Parameters

caption - *The message caption.*

message - *The message.*

public void info(String caption, String message)

Shows an info message on the window.

Parameters

caption - *The message caption.*

message - *The message.*

public void tray(String caption, String message)

Shows a tray message on the window.

Parameters

caption - *The message caption.*

message - *The message.*

}
