UCW Portal Dashboard and Gadget

We provide UCW IoT Cloud where you can run multiple instances of the **UCW Platform** for needs of your projects. You can start by registering here and get access to the IoT platform with unique functionality. To learn more check out the UCW Platform overview.

Tutorial overview

This is "Dashboard and Gadget" tutorial of a UCW portal add-on. Throughout the tutorial, you will build a dashboard. Since dashboard typically consists of few different types of gadgets, you will also implement them. Moreover, you will also learn how to store data in a data stream and create a scheduled job. You will also need knowledge from Hello World Tutorial. If you have not completed it yet, please see the tutorial pages. See source code.

Requirements

- Git
- Maven

Step 01: Download tutorial sources

To download tutorial sources and test their functionality, do as follows:

1. Make sure you have downloaded all the requirements.
2. Open a terminal/console and navigate to your working directory.
3. Download the sources using the following command:

```
git clone https://github.com/unitycloudware/ucw-portal-dashboard.git ucw-portal-dashboard
```

4. Move into newly create a folder

```
cd ucw-portal-dashboard
```

5. Checkout to a starting point of the tutorial

```
git checkout start
```

Note

In the tutorial, you will be working in "start" branch. However, at any moment of the tutorial, you can checkout into "master" branch to see correct solution.

6. Test downloaded files by building and running the sources. To do so, execute following commands

Unix / Linux based OS

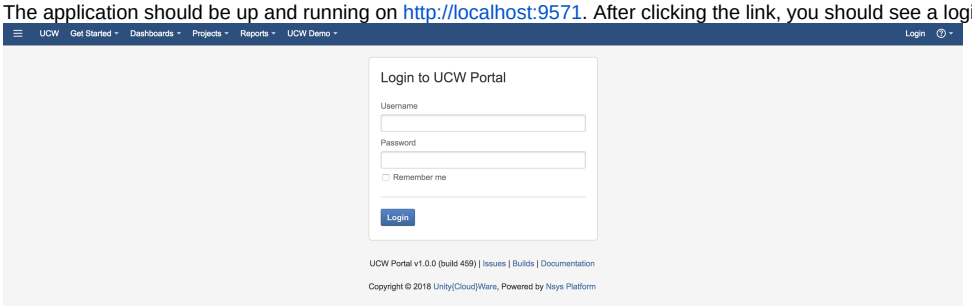
```
./build.sh
./run-portal.sh
```

or

Windows

```
build.bat
run-portal.bat
```

7. The application should be up and running on http://localhost:9571. After clicking the link, you should see a login page.



8. Log into the application using default credentials (User: admin, Password: admin) and admin page should be displayed.

You just setted essential environment for developing your first dashboard and gadget.

As you can see, there is a lot of stuff that has been already created. All of it has been created using UCW Template Module. If you have not familiarised with it yet, please see its documentation for more information.

Step 02: Add dashboard

Adding dashboard to a system is pretty simple. We just need few steps:

1. Open *nsys-plugin.xml* and paste following lines into it

```
<dashboard key="ucw-dashboard_dashboard" name="UCW Dashboard">
    <description>This dashboard will be empty.</description>
    <label>UCW Dashboard</label>
    <viewId>ucw-dashboard</viewId>
</dashboard>
```

2. Open *DashboardPlugin.java* and change PORTAL_REDIRECT and PORTAL_REDIRECT_DEFAULT variable with new ones:

```
public static final String PORTAL_REDIRECT = "nsys.portal.redirect";
public static final String PORTAL_REDIRECT_DEFAULT = "/ucw-dashboard";
```

3. Make use of new variables by changing input attributes of a *setProperty* function under the *customPortalConfig* method

```
customPortalConfig.setProperty(PORTAL_REDIRECT, PORTAL_REDIRECT_DEFAULT);
```

Note

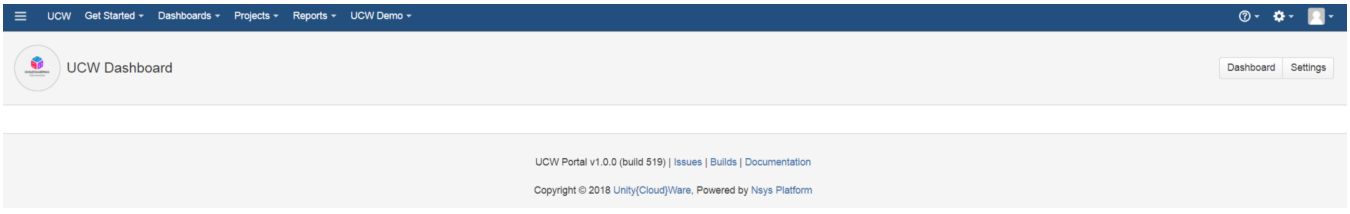
Theoretically, the first step would be enough to add a dashboard, but you would have to change the URL manually or add a link somewhere to the application. Step 2 and Step 3 makes the application to redirect you to the dashboard right after the login. It just requires you to set same value to property PORTAL_REDIRECT_DEFAULT as the value of *viewId* used in Step 1.

That's it. You just added your very first dashboard to the UCW plugin. Unfortunately, this dashboard has almost zero value since it shows no data. So, let's explain how to create a gadget containing some data and how to add it to the dashboard. Before that practice a bit how to add links to the top navigation menu, header actions or header logo (See Hello World Tutorial in case you do not know how to do it and also check Dashboard Plugin Module to see all available attributes of *dashboard* component). If you do not want to practice you can skip to Step 03.

Following code shows what needs to be done to complete the previous task. All steps need to be done only in *nsys-plugin.xml* file:

```
<navigation-section key="ucw-portal-tutorial_nav-main_demo" name="UCW Demo" location="system.top.navigation.bar
/ucw.demo" weight="1000">
    <label>UCW Demo</label>
    <description>UCW Demo Examples</description>
</navigation-section>

<navigation-item key="ucw-dashboard_nav-main_dashboard" name="UCW Dashboard" section="system.top.navigation.bar
/ucw.demo" weight="0">
    <label>UCW Dashboard</label>
    <link>/dashboard/view/ucw-dashboard</link>
    <conditions>
        <condition class="org.nsys.portal.conditions.UserIsLoggedInCondition" />
    </conditions>
</navigation-item>

<navigation-section key="ucw-dashboard_dashboard-header-actions" name="UCW Dashboard Header Actions" location="
ucw.tutorial.dashboard.header.actions" weight="0">
    <label>Dashboard Header Actions</label>
</navigation-section>

<navigation-item key="ucw-dashboard_dashboard-header-actions_dashboard" name="UCW Dashboard" section="ucw.
tutorial.dashboard.header.actions" weight="0">
    <label>Dashboard</label>
    <link>/dashboard/view/ucw-dashboard</link>
</navigation-item>

<navigation-item key="ucw-dashboard_dashboard-header-actions_settings" name="UCW Dashboard Administration"
section="ucw.tutorial.dashboard.header.actions" weight="0">
    <label>Settings</label>
    <link>/ucw-dashboard/admin</link>
</navigation-item>

<dashboard key="ucw-dashboard_dashboard" name="UCW Dashboard">
    <description>This dashboard provides an overview about UCW Dashboard.</description>
    <label>UCW Dashboard</label>
    <viewId>ucw-dashboard</viewId>
    <imageUri>${portalResourcesUrl}/resources/images/ucw_logo.png</imageUri>
    <actionButtons>ucw.tutorial.dashboard.header.actions</actionButtons>
</dashboard>
```

The result should look very similar to the to picture bellow



Step 03: Gadget minimum

In this step, you will learn how to add an empty gadget to the dashboard. First, you will create a gadget, and then you will use dashboard-gadget to add it to the dashboard. Follow the steps:

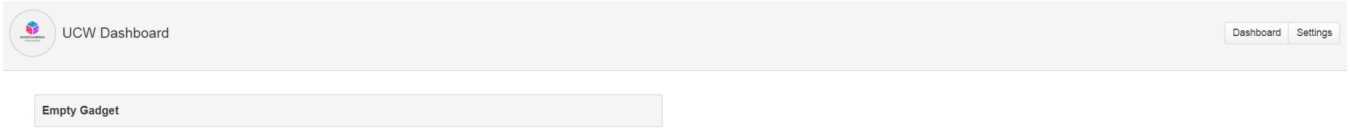
1. Create file EmptyGadget which extends AbstractGadget and paste following code into it

```
public static final String TEMPLATE = "/templates/empty-template.vm";

public DataMeasurementGadget() {
        setTemplate(TEMPLATE);
}

@Override
protected Map<String, Object> createVelocityParams(final Map<String, Object> context) {
    Map<String, Object> velocityParams = new HashMap<String, Object>();
    return velocityParams;
}
```

2. Create *empty-template.vm* file under templates directory
3. Add following code to *nsys-plugin.xml*

```
<dashboard-gadget key="ucw-dashboard_empty-gadget" name="Empty Gadget" class="com.unitycloudware.portal.
tutorial.dashboard.EmptyGadget">
    <description>Provides information about data measurement.</description>
    <label>Empty Gadget</label>
    <view>ucw-dashboard</view>
</dashboard-gadget>
```

That's it. You only need to extend AbstractGadget, set gadget template and implement the *createVelocityParams* function which has to return at least empty Map. When the gadget is created, add it to the dashboard. For more options of dashboard-gadget see its documentation. Resulting page should look as follows.



Similarly, as in case of empty dashboard empty gadget has almost zero value. If you have completed the Hello World Tutorial, you should have an idea how to add data to the gadget. You will need to use Apache velocity templates (see documentation) and append the data to Map returned from the *createVelocityParams* function. You can try it by yourself or follow the next steps which show few versions of gadgets.

Step 04: Add data measurement gadget

In this step, you will add gadget with randomly generated data,

1. Create DataMeasurementGadget class which extends AbstractGadget
2. Paste the following snippet into the class

```
public static final String TEMPLATE = "/templates/gadget/data-measurement.vm";

public DataMeasurementGadget() {
    setTemplate(TEMPLATE);
}

@Override
protected Map<String, Object> createVelocityParams(final Map<String, Object> context) {
    Map<String, Object> velocityParams = new HashMap<String, Object>();
    velocityParams.put("items", getData());
    return velocityParams;
}

protected List<DataItem> getData() {
    List<DataItem> items = new ArrayList<DataItem>();

    for (int i = 0; i < 10; i++) {
        items.add(DataItem.create(
                RandomRange.getRandomDouble(0, 100),
                RandomRange.getRandomInt(0, 100),
                TimeUtils.getNow().getTime()));
    }

    return items;
}
```

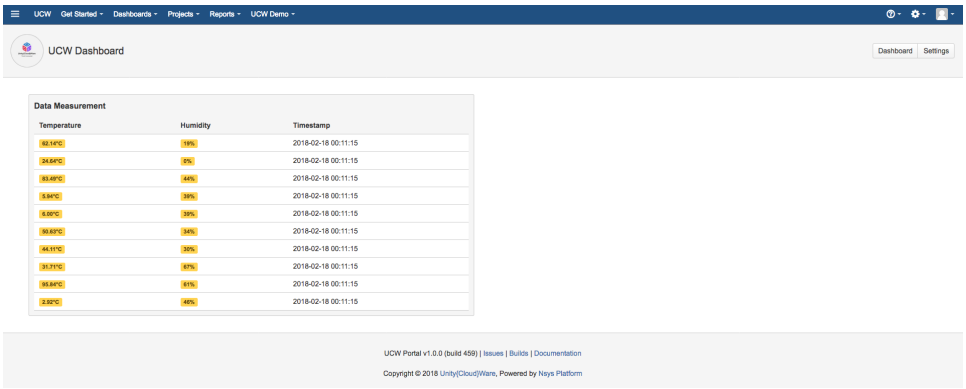
3. Create *data-measurement.vm* file inside resources folder and paste the following snippet into it

```
<table class="aui aui-table-interactive aui-table-sortable">
    <thead>
    <tr>
        <th id="data-temperature">Temperature</th>
        <th id="data-humidity">Humidity</th>
        <th id="data-timestamp">Timestamp</th>
    </tr>
    </thead>
    <tbody>
        #foreach ($item in $items)
        <tr>
            <td headers="data-temperature"><span class="aui-lozenge aui-lozenge-current">$String.format
("%.2f", $!{item.temperature})&deg;C</span></td>
            <td headers="data-humidity"><span class="aui-lozenge aui-lozenge-current">$!{item.humidity}%<
/span></td>
            <td headers="data-timestamp">$date.format("yyyy-MM-dd HH:mm:ss", $!{item.timestamp})</td>
        </tr>
        #end
    </tbody>
</table>
```

4. Register the gadget into a dashboard

```
<dashboard-gadget key="ucw-dashboard_data-measurement-gadget1" name="Data Measurement Gadget # 1" class="
com.unitycloudware.portal.tutorial.dashboard.gadget.DataMeasurementGadget">
    <description>Provides information about data measurement.</description>
    <label>Data Measurement</label>
    <column>left</column>
    <order>0</order>
    <view>ucw-dashboard</view>
</dashboard-gadget>
```

5. Build and run the application. You should see a page similar to the one shown in the picture below



That's it. The gadget shows data which were generated randomly in the DataMeasurementGadget.

Step 05: Add sensor data gadget

In this step, you will create a gadget which displays some dynamically generated data. Before you dive into it, try to create Job which will generate data into a data stream (User data stream name: "*ucw-dashboard*", project name: "*UCWD*" and device name: *"adafruit-feather-m0-wifi"*, also see UCW Job Scheduler tutorial in case you do not know how to do it. Note that you can always checkout to master branch to see the final version). When the job is created you can follow next steps.

1. Create SensorDataGadget class which extends AbstractGadget
2. Paste snippet into the class

```
private DeviceManager deviceManager;
private DataManager dataManager;

public static final String TEMPLATE = "/templates/gadget/sensor-data.vm";

public SensorDataGadget() {
    setTemplate(TEMPLATE);
}

public DeviceManager getDeviceManager() {
    if (deviceManager == null) {
        deviceManager = ComponentProvider.getInstance().getComponent(DeviceManager.class);
    }

    return deviceManager;
}

public DataManager getDataManager() {
    if (dataManager == null) {
        dataManager = ComponentProvider.getInstance().getComponent(DataManager.class);
    }

    return dataManager;
}

@Override
protected Map<String, Object> createVelocityParams(final Map<String, Object> context) {
    Map<String, Object> velocityParams = new HashMap<String, Object>();
    velocityParams.put("items", getData());
    return velocityParams;
}

protected List<DataItem> getData() {
    List<DataItem> data = new ArrayList<DataItem>();

    Device device = getDeviceManager().getDeviceByName(TestDataUtils.PROJECT_KEY, TestDataUtils.
DEVICE_NAME);

    if (device == null) {
        return data;
    }

    DataStream dataStream = getDataManager().getDataStream(TestDataUtils.PROJECT_KEY, TestDataUtils.
DATA_STREAM_NAME);

    if (dataStream == null) {
        return data;
    }

    // Load 10 last records
    List<DataStreamItem> items = getDataManager().loadStream(dataStream, device, 0, 10);

    if (items == null || items.isEmpty()) {
        return data;
    }

    for (DataStreamItem item : items) {
        // Process only data for data stream with type of DATA_MESSAGE
        if (item.getType() != DataStreamType.DATA_MESSAGE) {
            continue;
        }

        DataMessage dataMessage = (DataMessage) item.getData();

        // Transform JSON payload to DataItem object
        data.add(JsonUtils.fromJson(dataMessage.getData(), DataItem.class));
    }

    return data;
}
```

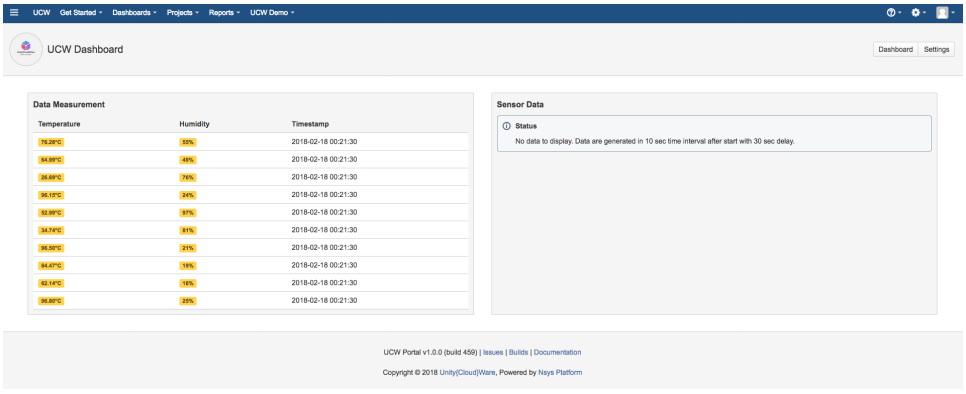
3. Create a *sensor-data.vm* file inside resources folder and paste the following snippet into it

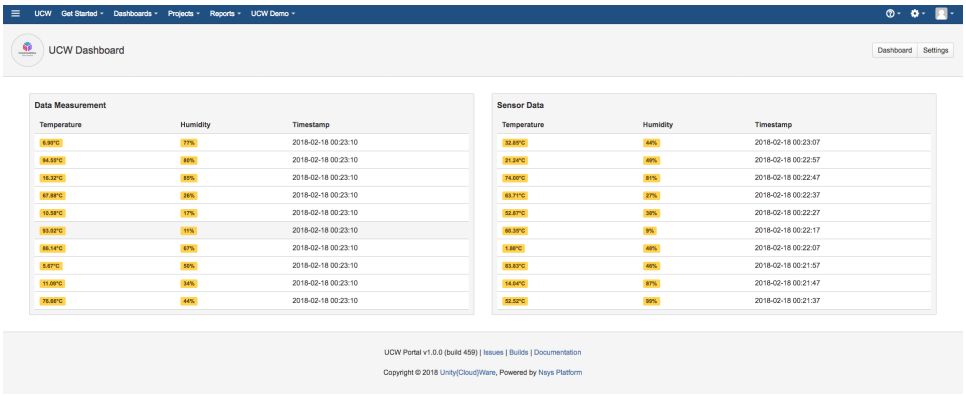
```
#parse ("macros.vm")

#if ($!{items.isEmpty()})
    #showMessage("info", "Status", "No data to display. Data are generated in 10 sec time interval after
start with 30 sec delay.", false)
#else
<table class="aui aui-table-interactive aui-table-sortable">
    <thead>
    <tr>
        <th id="data-temperature">Temperature</th>
        <th id="data-humidity">Humidity</th>
        <th id="data-timestamp">Timestamp</th>
    </tr>
    </thead>
    <tbody>
        #foreach ($item in $items)
        <tr>
            <td headers="data-temperature"><span class="aui-lozenge aui-lozenge-current">$String.format
("%.2f", $!{item.temperature})&deg;C</span></td>
            <td headers="data-humidity"><span class="aui-lozenge aui-lozenge-current">$!{item.humidity}%<
/span></td>
            <td headers="data-timestamp">$date.format("yyyy-MM-dd HH:mm:ss", $!{item.timestamp})</td>
        </tr>
        #end
    </tbody>
</table>
#end
```

4. Register the gadget into a dashboard

```
<dashboard-gadget key="ucw-dashboard_sensor-data-gadget" name="Sensor Data Gadget" class="com.
unitycloudware.portal.tutorial.dashboard.gadget.SensorDataGadget">
    <description>Provides information about sensor data measurement.</description>
    <label>Sensor Data</label>
    <column>right</column>
    <order>0</order>
    <view>ucw-dashboard</view>
</dashboard-gadget>
```

5. Build and run the application. The generator we have created for this tutorial generates data every 10 seconds, and so you should see pages similar to the ones shown in the pictures below

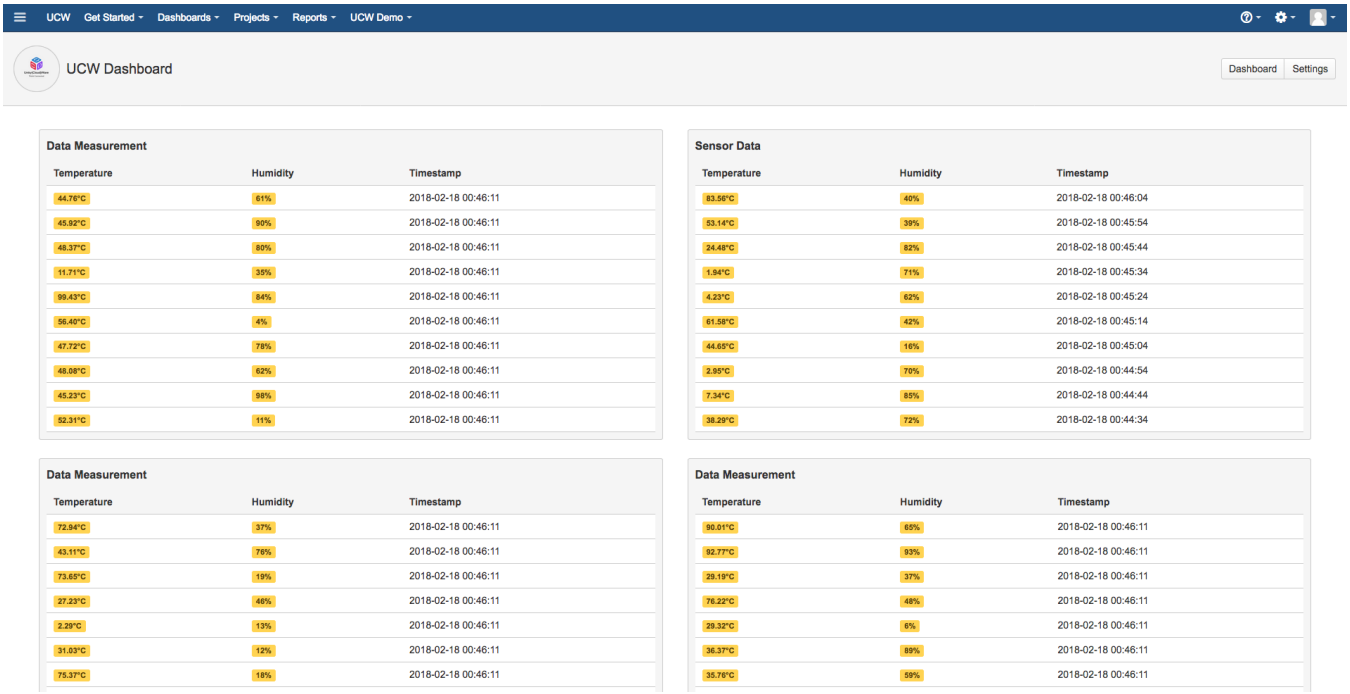




The gadget we have created shows sensor data for temperature and humidity stored in the data stream.

If you reach this part, you just created your first dashboard with multiple gadgets which can use another plugin and plugin functions

You should try to add few more gadgets to different positions, with different labels etc. to improve your understanding of gadget creation. Be aware of the key property you assign them (keys have to be unique). The result can look as follows.



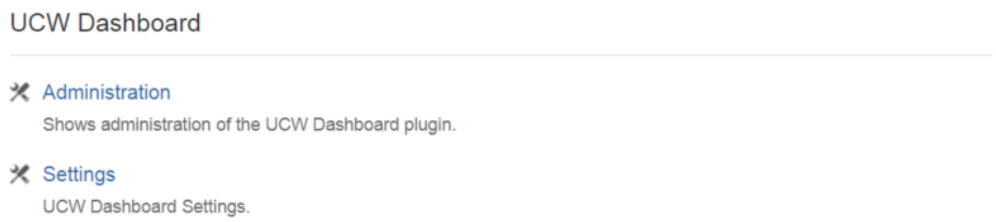
Step 06: Add admin gadget

If you access http://localhost:9571/admin, you should see a lot of links to pages with general information about the application, list of users and so on. This is the place where all plugins should register their administration interfaces. Let's assume we have to be able to set some value for our gadget which will affect its behaviour (for example seed of generator of random data). This variable can set only someone with administrator privileges and so we will need to create admin-gadget.

To create an admin gadget, we need almost the same steps as in case of creating a regular gadget. We will need template and class which extends AbstractGadget. Let's create them (See template and gadget files created for this tutorial) and register the gadget by using the following snippet:

```
<admin-gadget key="ucw-dashboard_admin-gadget_dashboard" name="UCW Dashboard Admin Gadget" class="com.
unitycloudware.portal.tutorial.dashboard.gadget.DashboardAdminGadget">
    <description>Provides an overview for the UCW Dashboard gadget.</description>
    <label>UCW Dashboard</label>
    <column>left</column>
    <order>2</order>
</admin-gadget>
```

If you build and run the application new section should appear in the administration page



Note

You can also add some links to sidebar menu of the administration page using navigation section and navigation item components. Check out the following example or refer to the nsys-plugin.xml:

```
<navigation-section key="ucw-portal-tutorial_admin-demo" name="UCW Demo Administration Navigation"
location="system.admin/ucw.demo" weight="90">
    <label>UCW Demo</label>
</navigation-section>

<navigation-item key="ucw-dashboard_admin-dashboard-settings" name="UCW Dashboard Settings" section="
system.admin/ucw.demo" weight="0">
    <label>UCW Dashboard Settings</label>
    <link>/ucw-dashboard/admin</link>
</navigation-item>
```

Note

If you checked out into the master branch of the tutorial both links should be working. However, if you stayed in branch start, both link will not be functional because you did not implement necessary controllers. Refer to the code to see how to do so or simply check out to master branch.