kayako

Example of Using “Hooks”

We are going to discuss in this example about adding a new tab in the “View Organization” page under Staff CP > Users > Manage Organization > open any organization. This tab will list all tickets that belong to an Organization.

Steps to follow :

1. Creating a custom app – You will need to create a custom app and hook it with the existing code. Place the custom app folder under the “__Apps” folder of your helpdesk installation.

2. Create a new folder in your custom app folder called “config”. The first required file is the “config.xml” file. This file contains the custom application information which is displayed under Admin CP > Apps > click on the respective app.

3. The second file required to be placed in the “Config” folder is the setup database file which has to be named matching the custom app name. For example, if the custom app folder is named “Organizationtickets”, this means our setup database file must be called “class.SWIFT_SetupDatabase_organizationtickets.php”. This file is very structured and contains some “key” functions within it, these functions control how Kayako installs, uninstalls and processes the application during upgrades. The function names we must create are “install”, “uninstall”, “construct”, “destruct” and finally “upgrade”.

4. Once we have created both the required files, we will see our application is listed under Admin CP > Apps section. The default state of the custom app will be “Not Installed”.

5. Next we need to create our “hook” into the Kayako system. To do this we need to create a special folder within our custom app folder called “hooks”. Within the hooks folder, we need to add a custom “hook” file, this file must be named to match the name of the Kayako defined hook we wish to use. As mentioned before, we are adding a tab to the user organization display page and therefore the file we need to create will be called “staff_userorganization_tabs.hook”. The code within this file is not part of using hooks, this code is actually the code we are adding to achieve our new functionality. The code in this file basically adds a new tab to the organization view and sets the tab contents to point to a new function we have not created yet.

6. As we are making a call to a function within our custom app which currently does not exist, so now we need to create the next folder within our custom app folder to enable us to create the required function. The reason we are creating a “staff” folder is because the change we are making is in the staff control panel, so when we call a function from our hook, it will automatically look for the controller within the “staff” folder of our custom application.

7. The file we need to create in “staff” folder will be called “class.Controller_organizationtickets.php”. In this file, we will have the function needed to display our new tab’s contents. The common elements within this file however will have to be the same as all other controllers. The file will need a construct and destruct function, along with our custom function.

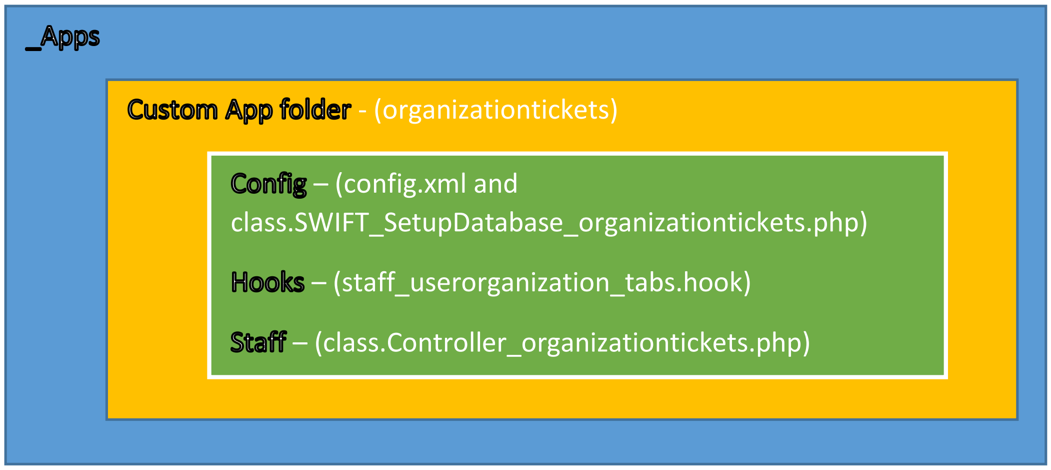
_Apps
Custom App folder - (organizationtickets)
Config – (config.xml and class.SWIFT_SetupDatabase_organizationtickets.php)
Hooks – (staff_userorganization_tabs.hook)
Staff – (class.Controller_organizationtickets.php)

Diagram illustrating above example and structure of files required to use “Hooks”.