(TcHmiTutorial) Multi Page - Single Server

2024.11.28

Concept

In some cases it may be interesting to have a single HMI server that has to send its data to multiple different screens, which all have their own data and settings.

This can be achieved simply by making a quick initialisation procedure which stores an ID in the browser local storage. Then the next time the page is opened, the framework will now which ID the screen has and can open the correct page for that particular screen.

This also works nicely in conjunction with kiosk mode on chrome. When the process has an explicit data directory, then each browser instance can have their own ID saved in storage. On load, the HMI framework can respond to this.

Details

Chrome

Starting chrome in kiosk mode is quite easy. This can be set as a startup process, or from a shortcut. In this case we will look at the shortcut option. In this case there are two shortcuts

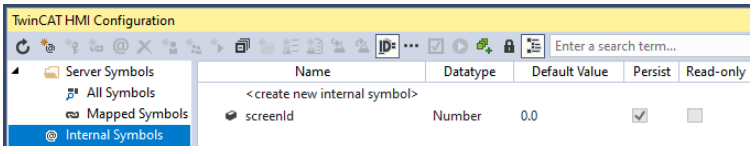
Where the first shortcut points to `"C:\Program Files (x86)\Google\Chrome\Application\chrome.exe" --kiosk --user-data-dir="C:\hmi_data\chrome_data 1" "http://localhost:2010/"`
And the second points to `"C:\Program Files (x86)\Google\Chrome\Application\chrome.exe" --kiosk --user-data-dir="C:\hmi_data\chrome_data 2" "http://localhost:2010/"`

So they will each start in Kiosk mode, open the exact same HMI endpoint (in this case `http://localhost:2010`), but will have their own data directory. It is possible to build a reset procedure into the HMI for this, so the local storage will be cleared. But in extreme cases you can also simply delete the corresponding storage directory. This way the Chrome process will reinitialise itself at next boot.

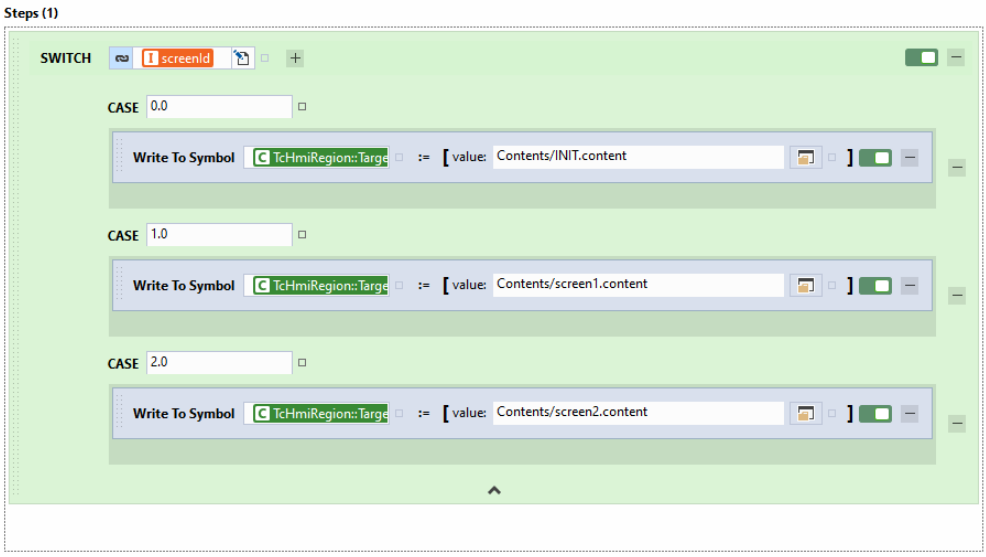
The HMI

A simple example project is available at: `...`

This project consists of a single region that will be filled dynamically with some content. The dynamic part will depend on an internal variable (which will be stored in the browser local storage). This variable is set to 0 initially:

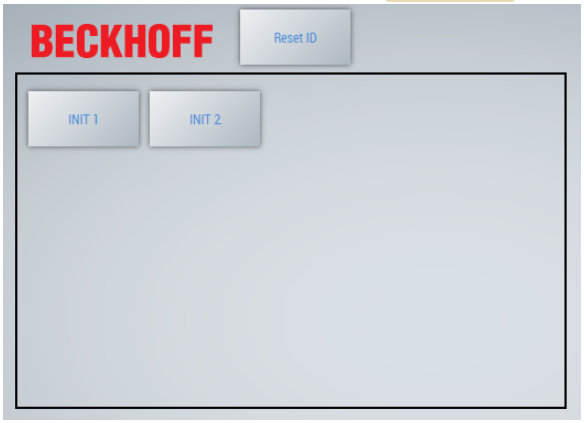


When the region loads, on the `.OnAttached` event, the region will respond to the Internal Symbol:

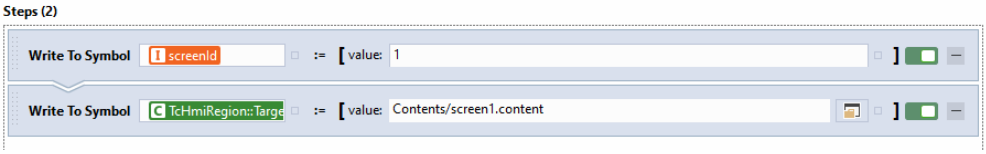


Which will load the corresponding page when the region is loaded.

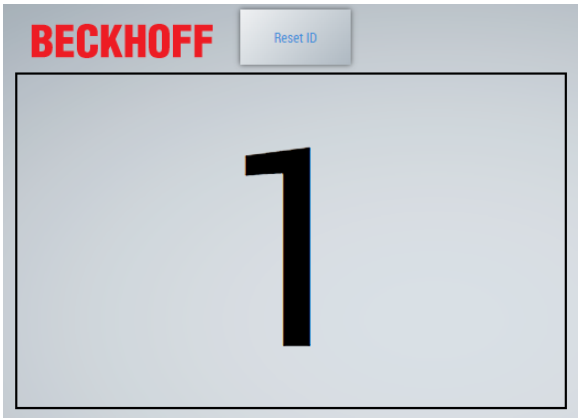
When no selection has been made yet, the `INIT.content` page is loaded which allows the user to select which screen ID should be set:



When one of the buttons is pressed it will set the ID in the local storage and switch to the corresponding page:



Which will result in:

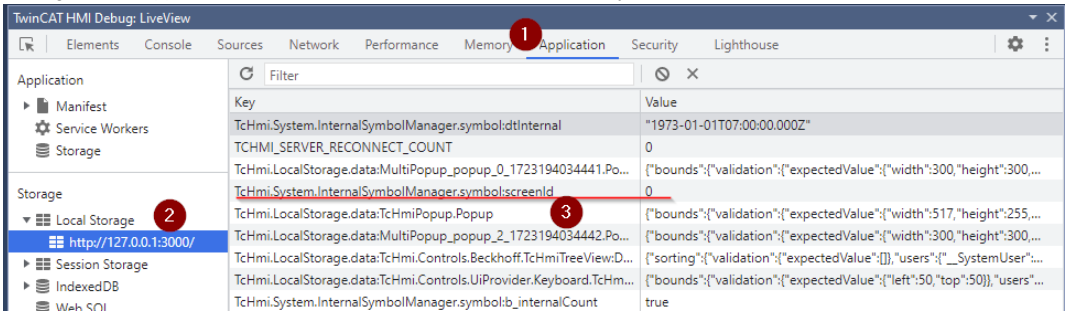


Now when the page is reloaded or the browser is restarted, the event on the region will select the corresponding page as selected on the INIT page.

The reset button will set the internal variable back to '0', and switch back to the `INIT.content` page.

To check the function of the local variable you can open the Developer Tools in the Live View or in the browser (not available in Kiosk mode)

Then go to the 'Application' tab to see the status of the internal memory for that browser instance.



Questions and troubleshooting

Why do you need the local storage/internal variables

The browser does not easily get access to system parameters, this is an explicit security design choice that is difficult and dangerous to subvert. Therefore you need to do this either on browser level (by storing an ID in the persistent browser cache). Or by handling this in advance in the server, which would entail more complexity.

The local storage seems not to be persistent

First make sure that the local storage is actually set by checking the browser console. Then make sure it is changed after a reload. The Internal Variable you use must be marked 'Persistent'.

I made a mistake and want to reset the local storage

This can be hard on kiosk instances of a browser. The easiest way (if implemented as provided in this document) is to delete the Chrome Data Directory. Then the system will go through the init procedure on startup. If you want more control, you can build an explicit reset procedure into the HMI page. For example setting all Internal Variables back to their initial values.

Testing local storage

There is a short example project available in the git repository `localStorageExample/index.html`. This example makes it possible to quickly test the working of local storage without the complexities of the HMI Framework. Simply navigate to this file in your browser to test it out. Using the browser console (`F12` in most browsers) to explore the working of local storage. The TwinCAT HMI uses this in the background for internal variables that have been marked 'Persist'.