Emulation Behavior Tasks

ApplinX HTML Emulation provides a ready to use, fully functional Web emulation. It is part of ApplinX installation and is available in JSP and .NET. It is a thin-client Web application and uses JavaScript and HTML for configuration and fine-tuning.

To install a new HTML emulation refer to the Framework Manager or to the Eclipse Plug-in/Visual Studio documentation. The emulation template is based on the new_application Web application. It is possible to configure various features in the Framework Configuration Editor in the Emulation node. All the emulation features are relevant both for instant and generated pages (that inherit from GXDefaultLogicContext) unless otherwise displayed in the Emulation node in the Framework Configuration Editor. Detailed below are some tasks which require configuring a number of parameters.

- Customizing the Background Check for Host Screen Changes
- Enabling the User to Control the Font Size
- Opening Multiple Web Sessions
- Printing a Capture of the Host Screen
- Enabling Sending Dup and FieldMark Characters to the Host

Customizing the Background Check for Host Screen Changes

Use the user exits in *userExit.js*.

GXHostScreenChecker.hostScreenSeqCheckBeforeRefresh(isDirty) > returns boolean

This user exit method is used to perform additional queries for this feature. For example to check whether the data on the screen is saved. If it is not saved, return false to stop the browser from refreshing to prevent data loss.)

Example:

```
/**
 *    userExists.js     prompt user if the screen is dirty
 */
function GXHostScreenChecker.hostScreenSeqCheckBeforeRefresh(isDirty) {
      //check
      if (isDirty) {
            //ask & get answer from user...
                        alert("Host screen was updated. Data entered will be lost");
            return false; //stop refreshing
      }
      return true; //continue refreshing
}
```

GXHostScreenChecker.hostScreenSeqCheckSetTimer(ticks) > returns times

This user exit method is used to change the times for the checker timer. The user gets the current setting time and can return a different value. For example: the user finds 3, 6, 12, 24 too frequent, so in order to increase the time intervals he can return time*3 which results in 9, 18, 36, 72)

Example:

```
/**
 *    userExists.js     reset the timer
 */
function GXHostScreenChecker.hostScreenSeqCheckSetTimer(ticks) {
      return ticks*3;
}
```

Enabling the User to Control the Font Size

It is possible to enable the user to control the font size used in the Web application. Font sizes between 10 and 24px are possible (10, 12, 14...24px).

To enable user control of the font size:

1. Copy the plus/minus links from the header section in the HTML emulation.
2. Set these links to call the `gx_increaseFontSize()` or the `gx_changeFontSize(<desired size in pixels>)`JavaScript functions and the `gx_decreaseFontSize()` JavaScript function.

   Example:

   ```
   <input type="button" value="+" onclick="gx_increaseFontSize(12);" />
   ```

3. Refer to Configuring your Framework and access the Framework Configuration Editor. In the **Instant** node, **Font size** field, select the **Dynamic by resolution** option from the drop-down list.

Refer to the API:

- `gx_increaseFontSize`
- `gx_decreaseFontSize`
- `gx_changeFontSize`

Opening Multiple Web Sessions

A known problem of Web servers is that it is not possible to open multiple Web sessions (against the same Web application) from the same Web window (in a portal application for example). The ApplinX session is dependant on the web session, therefore it is problematic to open multiple host sessions using ApplinX Framework. To overcome this problem ApplinX provides an ActiveX component, which creates a new IE process whenever a new Web window is required.

In *index.aspx/jsp* (in the emulation_template folder) uncomment the following line:

```
<OBJECT ID="gx_emulationComponent" CLASSID="CLSID:FE93BC5E-332E-41D7-9B36-EA36265998CA" CODEBASE="z_lib\GXOsApi.CAB#version=1,0,0,0"></OBJECT>
```

Note:
The following function already exists in the *index.aspx/jsp* file. Notice the use that the function does with the "gx_emulationComponent" object.

```
function connect(){
        var newUrl = "z_resourceReader.aspx?res=pages/z_openFull.htm&gx_page=gxfirstpage.aspx";
        if (window.gx_emulationComponent){
                // If the ActiveX was loaded use it to open a new
  // Internet Explorer window with its own IE process.
                gx_openNewBrowser(newUrl);
        }
        else{
                // If the ActiveX wasn’t enabled open a window normally
                location.href=newUrl;
        }
}
```

Printing a Capture of the Host Screen

As in most terminal emulators, also ApplinX enables printing a snapshot of the current host screen. By executing a JavaScript function a pop-up screen appears enabling users to print the current screen. This window, unlike the Instant screen, does not include any changes (such as transformations) in the screen.

To enable users to print a capture of the host screen:

- Add a button/link to the application’s Master page (*template.jsp/template.master.cs*). You can place it anywhere you see fit. Set it to call `gx_printScreen()`.

Example

```
<input type="button" id="printButton" onClick="gx_printScreen();" />
```

Refer to the API:

- `gx_printScreen`

Enabling Sending Dup and FieldMark Characters to the Host

Dup and FieldMark are special mainframe characters. In instant and generated pages, a user can send to the host a not printable character in specific input fields.

To enable sending Dup and FieldMark characters to the host:

1. Refer to Configuring your Framework and access the Framework Configuration Editor. In the **Emulation** node, select the **Support Dup and FieldMark host keys** check box.

2. In the **Keyboard mapping** node, map two keys, for example:

   Host action key: [dup]; Press a keyboard combination: CTRL+D

Host action key: [fieldmark]; Press a keyboard combination: CTRL+F

3. You can also perform this mapping using JavaScript:

```
function pageOnLoad(){
     gx_engine.addKeyBoardMapping(
           GXAdditionalKey.CTRL,
           68 /*  Character ’d’  Ascii code */  ,
           gx_dup(),
           true);
}
```

4. When pressing CTRL+D (dup) the field will be marked with an asterisk ("*") and the cursor will move to the next field.

   When pressing CTRL+F (fieldMark) the field will be marked with a semi colon (";").

Refer to the API:

- `gx_fieldmark`
- `gx_dup`