Instant Pages Customization

Using a Proportional (Non-Fixed) Font in Web Pages

Controlling Instant Display Properties

Code Transformations

- Creating a New Code Transformation
- Applying a New Transformation to all Screens
- Applying a New Transformation to a Screen Group

Manipulating Individual Host Fields

- Positioning Specific Fields
- Formatting Specific Fields
- Replacing a Field’s Text
- Replacing a Field with a Web Element, Adding a Web Element

Manipulating Host Characters

Manipulating Host Keys

Improving Transitions between Screens

Using a Proportional (Non-Fixed) Font in Web Pages

By default, ApplinX uses a monospaced, fixed font (Courier New) for displaying instant HTML pages. However, it is possible to use a proportional (non-fixed) font, to achieve a more modern, Web-like look & feel. Since the rendering of the HTML screen is based on absolute positioning, using a proportional font usually does not damage the correct alignment of the host fields (although some data may appear misaligned).

To use a proportional font in Web pages (JSP & .NET):

-

  For Instant HTML Pages

  In the file *\css\styles_instant.css*, change the following classes to use a proportional font of your choice, for example:

```
#gx_screenArea
{
         position:relative;
         font-family: "Verdana";
         height:0px; /* make scrolling when needed*/
}
#gx_screenArea input,
#gx_screenArea select {
         font-family: "Verdana";
         position:absolute
}
```

Or:

For Generated Pages

In the file *\css\styles_generated.css*, in a similar manner as for instant HTML pages, change all the classes that use the font "courier new" to a proportional font.

Controlling Instant Display Properties

It is possible to set various display properties of instant HTML pages, such as colors, fonts, etc. This is usually done in order to apply a Web look & feel to the ApplinX Web application.

Most of these settings are done in the style sheet file. Specific settings are set in other files. Refer to the relevant tasks for more details.

To control instant display properties (JSP and .NET):

1. The *css\styles_instant.css* file contains all CSS classes used for rendering instant HTML pages. Change them as necessary to affect the colors and fonts used for the pages.

2. To find out which CSS class is used for a specific element or field in the instant HTML page, it is possible to use the Browser view source option to view the generated HTML source, and then look for the element and extract its CSS classes.

   Note:
   This style sheet applies only to instant HTML pages. For modifying the styles used for generated web pages (JSP, ASPX), use the *\css\styles_generated.css* file.

Code Transformations

A Code Transformation (or Code Transform) is a code class (written in Java, C# or VB.NET for JSP or .NET frameworks respectively), that is executed on all host screens or on a specific group of screens (i.e. an ApplinX Screen Group) and modifies the rendered instant HTML page for the current host screen it is executed on.

Refer to

- Creating a New Code Transformation

- Applying a New Transformation to all Screens
- Applying a New Transformation to a Screen Group
- Manipulating Individual Host Fields
- Positioning Specific Fields
- Formatting Specific Fields
- Replacing a Field’s Text
- Replacing a Field with a Web Element, Adding a Web Element
- Manipulating Host Characters
- Manipulating Host Keys

Creating a New Code Transformation

In most cases, transformations are defined using the Transformation Wizard . However, when due to limited flexibility, it is not be possible to define certain required transformations using the wizard, complement the wizard-defined Transformation entities with code-defined Transformation classes.

Creating a new Instant transformation is carried out by using the provided template transformations (`UserTagTransform1`, `UserCompletionTransform1`, `UserHostKeysTagTransform`).

`UserTagTransform1` contains events for each individual rendering event and should be used for new transformations that manipulate specific tags (input fields, output fields, GUI elements, etc.) - for example, adding images or repositioning fields.

`UserCompletionTransform1` is called upon completion of the entire rendering process and should be used for new transformations that manipulate the entire rendered HTML - for example, a transformation that converts a host menu screen to a list of hyperlinks.

To create a new transformation (JSP):

1. Copy one of the template transformations from *\web-inf\classes\transforms* and rename the file and the class:

   ```
   public class SampleTransform extends GXTagListener{
   ```

2. Add code in the relevant stub methods.
3. To apply the transformation, see the tasks Applying a New Transformation to all Screens and Applying a New Transformation to a Screen Group.

To create a new transformation (.NET):

1. Copy one of the template transformations from the *transforms* directory and rename the file and the class:

```
public class SampleTransform: GXTagListener
```

2. Add code in the relevant stub methods.
3. To apply the transformation, see the Applying a New Transformation to all Screens and Applying a New Transformation to a Screen Group.

Applying a New Transformation to all Screens

This task applies an existing transformation. To create a new transformation, see the task Creating a New Transformation.

To apply a new transformation to all host screens (JSP):

- In the file *web-inf\classes\contexts\GXInstantLogicContext.java*, uncomment the code in the function `registerInstantTransforms` and register the new transformation class:

```
public void registerInstantTransforms()    {
             GXRenderConfig instantConfig = getGXAppConfig().getInstantConfig();
             instantConfig.addTagListener(new UserTagTransform1());
  // add here more transform registrations
}
```

`UserTagTransform1` being the name of the new transformation class.

To apply a new transformation to all host screens (.NET):

- In the file *GXInstantLogicWebForm.cs*, uncomment the code in the function `registerInstantTransforms` and register the new transformation class:

```
public virtual void registerInstantTransforms() {
                gx_appConfig.InstantConfig.addTagListener(new UserTagTransform1());
                // add here more transform registrations
}
```

`UserTagTransform1` being the name of the new transformation class.

Applying a New Transformation to a Screen Group

This task applies an existing transformation. To create a new transformation, refer to Creating a New Transformation.

It is possible to apply a transformation only for a specific Screen Group (or several Screen Groups). For example, a transformation that handles the command line field should be applied to the screen group of all screens containing this field.

To apply a new transformation to a screen group (JSP):

1. If the screen group has its own generated Web page, add the registration function to the file *web-inf\classes\contexts\screenGroupName.java*, to the function `registerInstantTransforms:`

```
public void registerInstantTransforms() {
super.registerInstantTransforms();
GXRenderConfig instantConfig = getGXAppConfig().getInstantConfig();
                instantConfig.addTagListener(new MyTagTransform());
            instantConfig.addCompletionListener(new MyCompletionTransform());
}
```

Where `MyTagTransform` is the name of the new transformation class and `MyCompletionTransform` is the name of a new completion transformation class.

2. If the screen group does not have its own generated Web page, register it in *web-inf\classes\contexts\GXInstantLogicContext.java*, and add code that will register it only for the appropriate screen group:

```
public void registerInstantTransforms() {
GXRenderConfig instantConfig = getGXAppConfig().getInstantConfig();
try {
if (getGXSession().getScreen().isMemberOf("MyScreenGroup")) {
        instantConfig.addTagListener(new MyTagTransform());
        instantConfig.addCompletionListener(new MyCompletionTransform());
}
} catch (GXGeneralException e) {}
}
```

Where `MyScreenGroup` is the name of the screen group, `MyTagTransform` is the name of a new tag transformation class and `MyCompletionTransform` is the name of a new completion transformation class.

3. It is also possible to add code in the transformation class itself that will activate it only for the appropriate screen group(s).

To apply a new transformation to a screen group (.NET):

1. If the screen group has its own generated Web page, add the registration function to the file *screenGroupName.aspx.cs*, to the function `registerInstantTransforms`:

```
public override void registerInstantTransforms() {
base.registerInstantTransforms();
gx_appConfig.InstantConfig.addTagListener(new MyTagTransform ());
gx_appConfig.InstantConfig.addCompletionListener(new MyCompletionTransform());
}
```

Where `MyTagTransform` is the name of the new transformation class and `MyCompletionTransform` is the name of a new completion transformation class.

2. If the screen group does not have its own generated web page, register it in *web-GXInstantLogicWebForm.cs*, and add some code that will register it only for the appropriate screen group:

```
public virtual void registerInstantTransforms() {
if (gx_session.getScreen().isMemberOf("MyScreenGroup")) {
                      gx_appConfig.InstantConfig.addTagListener(new MyTagTransform ());
gx_appConfig.InstantConfig.addCompletionListener(new MyCompletionTransform());
}
}
```

Where `MyScreenGroup` is the name of the screen group, `MyTagTransform` is the name of a new tag transformation class and `MyCompletionTransform` is the name of a new completion transformation class.

3. It is also possible to add code in the transformation class itself that will activate it only for the appropriate screen group(s).

Manipulating Individual Host Fields

The following tasks handle manipulation of specific host fields. The common methodology for such manipulation is identifying the screen groups including these fields, mapping the relevant fields as application fields and writing custom transformations for handling the mapped fields. See also Instant Web Application Development Methodology.

Note:
It is possible to carry out basic manipulation of fields using the Transformation Wizard.

Positioning Specific Fields

It is possible to reposition a specific field in a different position in the Web page (instead of in its original host position). It is also possible to display a field in one of the template sections.

Note:
It is recommended to map the fields as application fields. Refer to the Instant Web Application Development Methodology section for general instructions on mapping fields to screen groups.

Note:
It is possible to carry out basic manipulation of fields using the Transformation Wizard, Text to Text transformation or Input Field to Text Field transformation (detailed in the Transformations).

To position a specific field:

1. To reposition a field, create a new tag transformation and register it to the relevant screen group(s) or to all screens. Refer to Creating a New Transformation, Applying a New Transformation to all Screens and Applying a New Transformation to a Screen Group.

2. In the transformation class, add code that will reposition the field in the appropriate method. It will usually be `onNewLabel` (for output fields) or `onNewTextField` (for input fields):

   JSP

   ```
   public void onNewLabel(GXRenderEvent event, GXILabelTag label) {
   if (label.getId().equalsIgnoreCase("MyAppField")) {
                           label.setPosition(new com.sabratec.util.GXPosition(3,30));
           }
   }
   ```

   Where `MyAppField` is the name of the mapped application field to reposition and 3,30 is the new position, in host units.

   .NET

```
public override void onNewLabel(GXRenderEvent e, GXILabelTag label) {
if (label.getId() == "MyAppField") {
                        label.setPosition(new com.sabratec.util.GXPosition(3,30));
             }
}
```

Where MyAppField is the name of the mapped application field to reposition and 3,30 is the new position, in host units.

Formatting Specific Fields

It is possible to display specific host fields in specific styles. For example, displaying the message line field in a large, red font. This is done using the style sheet (CSS) classes.

Note:
It is recommended to map the fields as application fields. Refer to the Instant Web Application Development Methodologysection for general instructions on mapping fields to screen groups.

Note:
It is possible to carry out basic manipulation of fields using the Transformation Wizard, Text to Text transformation or Input Field to Text Field transformation (detailed in the Transformations).

To format a specific field:

1. Create a new tag transformation and register it to the relevant screen group(s) or to all screens. Refer to Creating a New Transformation, Applying a New Transformation to all Screens and Applying a New Transformation to a Screen Group.

2. In the transformation class, add code that will reposition the field in the appropriate method. It will usually be `onNewLabel` (for output fields) or `onNewTextField` (for input fields). The following code demonstrates replacing a field with a hyperlink:

   JSP

   ```
   public void onNewLabel(GXRenderEvent event, GXILabelTag label) {
   if (label.getId().equalsIgnoreCase("MyAppField")) {
   label.removeAttribute("class"); //Needed for removing all css classes
                           label.setAttribute("class","MyCSSClass");
   }
   }
   ```

   Where MyAppField is the name of the mapped application field to format and MyCSSClass is the name of the CSS class that contains the required formatting.

   .NET

   ```
   public override void onNewLabel(GXRenderEvent e, GXILabelTag label) {
   if (label.getId() == "MyAppField") {
   label.removeAttribute("class"); //Needed for removing all css classes
                           label.setAttribute("class","MyCSSClass");
              }
   }
   ```

   Where MyAppField is the name of the mapped application field to format and MyCSSClass is the name of the CSS class that contains the required formatting.

Replacing a Field’s Text

It is possible to replace the original host text of a specific field with other text.

Note:
It is recommended to map the fields as application fields. Refer to the Instant Web Application Development Methodologysection for general instructions on mapping fields to screen groups.

Note:
It is possible to carry out basic manipulation of fields using the Transformation Wizard, Text to Text transformation.

Create a new tag transformation and register it to the relevant screen group(s) or to all screens. Refer to Creating a New Transformation, Applying a New Transformation to all Screens and Applying a New Transformation to a Screen Group.

In the transformation class, add code that will replace the field’s text. It will usually be `onNewLabel` (for output fields) or `onNewTextField` (for input fields):

JSP

```
public void onNewLabel(GXRenderEvent event, GXILabelTag label) {
if (label.getId().equalsIgnoreCase("MyAppField")) {
                       String text = label.getContent().trim();
                if (text.equalsIgnoreCase("HostText1")) {
                         label.setText("NewText1");
                } else if (text.equalsIgnoreCase("HostText2")) {
                         label.setText("NewText2");
                }
                //...
}
```

Where `MyAppField` is the name of the mapped application field that its text is to be replaced, `HostText1, HostText2` are two original host texts and `NewText1, newText2` are two new texts.

.NET

```
public override void onNewLabel(GXRenderEvent e, GXILabelTag label) {
if (label.getId() == "MyAppField") {
                       String text = label.getContent().Trim();
                if (text == "HostText1") {
                         label.setText("NewText1");
                } else if (text == "HostText2") {
                         label.setText("NewText2");
                }
                //...
            }
}
```

Where `MyAppField` is the name of the mapped application field that its text is to be replaced, `HostText1, HostText2` are two original host texts and `NewText1, newText2` are two new texts.

Replacing a Field with a Web Element, Adding a Web Element

It is possible to replace a specific field with a Web element such as a button, a hyperlink, an image etc.

Note:
It is recommended to map the fields as application fields. Instant Web Application Development Methodology section for general instructions on mapping fields to screen groups.

Note:
It is possible to carry out basic manipulation of fields using the Transformation Wizard, Text to Hyperlink transformation or Text to Image transformation or Text to Button transformation.

To replace a field with a Web element:

1. Create a new tag transformation and register it to the relevant screen group(s) or to all screens. Refer to Creating a New Transformation, Applying a New Transformation to all Screens and Applying a New Transformation to a Screen Group.

2. In the transformation class, add code that will reposition the field in the appropriate method. It will usually be `onNewLabel` (for output fields) or `onNewTextField` (for input fields). The following code demonstrates replacing a field with a hyperlink:

   JSP

   ```
   public void onNewLabel(GXRenderEvent event, GXILabelTag label) {
   if (label.getId().equalsIgnoreCase("MyAppField")) {
   GXILinkTag myLink = com.sabratec.applinx.presentation.tags.html.GXHtmlTagFactory.instance().newLink("Software AG");
                           myLink.setTarget("http://www.softwareag.com/");
                           event.getScreenTagModel().replace(label, myLink);
           }
   }
   ```

   Where `MyAppField` is the name of the mapped application field to replace.

   .NET

   ```
   public override void onNewLabel(GXRenderEvent e, GXILabelTag label) {
   if (label.getId() == "MyAppField") {
   GXILinkTag myLink = com.sabratec.applinx.presentation.tags.html.GXHtmlTagFactory.instance().newLink("Software AG");
                           myLink.setTarget("http://www.softwareag.com/");
                           e.getScreenTagModel().replace(label, myLink);
           }
   }
   ```

   Where `MyAppField` is the name of the mapped application field to replace.

Manipulating Host Characters

It is possible to manipulate host characters, for example, removing unnecessary characters such as dots (.) or dashes (-), and replacing them with other text or HTML elements, etc.

Note:
It is possible to carry out basic manipulation of fields using the Transformation Wizard, Text to Text transformation.

To handle host characters:

- Create a new tag transformation and register it to the relevant screen group(s) or to all screens. Refer to Creating a New Transformation, Applying a New Transformation to all Screens and Applying a New Transformation to a Screen Group.

  In the transformation class, add code that will manipulate the host characters as required. It will usually be `onNewLabel` (for output fields) or `onNewTextField` (for input fields). The following code demonstrates removing dashes (-):

  JSP

  ```
  public void onNewLabel(GXRenderEvent event, GXILabelTag label) {
  if (label.getContent().indexOf("--") > -1) {
                                String text = label.getText();
                         text = com.sabratec.util.GXStringUtil.replaceAll(text,"-","");
                         label.setText(text);
          }
  }
  ```

  .NET

  ```
  public override void onNewLabel(GXRenderEvent e, GXILabelTag label) {
  if (label.getContent().IndexOf("--") > -1) {
  String txt = label.getText();
                          txt = com.sabratec.util.GXStringUtil.replaceAll(txt,"-","");
                          label.setText(txt);
               }
  }
  ```

Manipulating Host Keys

It is possible to change the way host keys are displayed, add additional host keys and remove host keys.

To manipulate host keys

- Refer to the commented sample code in any ApplinX new application, UserHostKeysTagTransform.java/cs/vb.

Improving Transitions between Screens

Wrapping the `GXPlaceHolder` component in the template page with a Partial Page rendering capablility control such as a gx:div, will reduce the traffic between the web application server and the user client , since only the Instant part of the page will be transferred rather than the whole page. This will result in a slightly better performance and smoother transition between screens.

JSP:

1. In the *Template.JSP* file, find the following control:

   ```
   <gx:placeholder id="GXPagePlaceHolder">Design time page content</gx:placeholder>
   ```

2. Place a `gx:div` tag around the `GXPagePlaceHolder` :

   ```
   <gx:div id="instantPanel">
               <gx:placeholder id="GXPagePlaceHolder">Design time page content</gx:placeholder>
    </gx:div>
   ```

3. Override the default behavior of each PF button on your keyboard. In *js/userExits.js*, change the `globalOnKeyDown` function to be of the following structure, where the if statement is before the `activateifexists` JavaScript code:

```
function globalOnKeyDown(gx_event){
        // use win.<SOMETHING> to access the page tags
        // for example: win.document.GX_form
        var win = gx_event.window;
        // if the key a PF key or ENTER
        if ((gx_event.keyCode>111 && gx_event.keyCode<124) || gx_event.keyCode==13 ){
                    // Update only the instantPanel part of the page
                    gx_updatePagePart('instantPanel');
        }
        // activate page scope function if exits
        activateIfExists(gx_event,gx_event.window.pageOnKeyDown); //gx_event.cancel();
        // for cancel the event
}
```

.NET:

1. In the *Template.master* file, find the following control:

```
<asp:ContentPlaceHolder ID="GXPagePlaceHolder" runat="server"/>
```

2. Place a `div` tag around the `GXPagePlaceHolder`:

```
<div
        runat="server" id="instantPanel"> <asp:ContentPlaceHolder
        ID="GXPagePlaceHolder" runat="server"/>
</div>
```

3. Override the default behavior of each PF button on your keyboard. In *js/userExits.js*, change the `globalOnKeyDown` function to be of the following structure, where the if statement is before the `activateifexists` JavaScript code:

```
function globalOnKeyDown(gx_event){
        // use win.<SOMETHING> to access the page tags
        // for example: win.document.GX_form
        var win = gx_event.window;
        // if the key a PF key or ENTER
        if ((gx_event.keyCode>111 && gx_event.keyCode<124) || gx_event.keyCode==13 ){
                    // Update only the instantPanel part of the page
                    gx_updatePagePart('instantPanel');
        }
        // activate page scope function if exits
        activateIfExists(gx_event,gx_event.window.pageOnKeyDown); //gx_event.cancel();
        // for cancel the event
}
```