Introduction to Eclipse Rich Client Platform Support in IBM Rational HATS

For IBM System i (5250)
Lab instructions

This lab teaches you how to use IBM Rational HATS to create a rich client plug-in application capable of transforming your 5250 applications.

Please report any issues or problems with this lab to Will Smythe (smythew@us.ibm.com).

Learning objectives

♦ Use the HATS Toolkit to create a HATS rich client project
♦ Customize a HATS template
♦ Record a simple sign on macro
♦ Transform a screen

Total time: 2 hours

Skill level

Introductory

Audience

Programmer

Requirements

♦ IBM Rational Application Developer 7.0.0.5 (or above)
♦ IBM Rational Host Access Transformation Services (HATS) 7.1 (or above)
♦ Internet connection

Prerequisites

♦ Basic IDE skills
♦ Basic understanding of terminal applications

Conventions used in this lab

♦ **Bold font** for user interface controls
♦ Mono-spaced font for user input and code blocks
♦ *Italic font* for variable names and glossary terms
HATS rich client platform overview

The following figure shows the stages of development, deployment, and runtime for a HATS rich client application.

1. The developer uses the HATS Toolkit to develop and test a HATS rich client application.

2. When ready, the developer exports the HATS application as an Eclipse feature, creates an update site containing the application feature, and uploads it to a Web server for deployment.

   **Note:** Different methods can be used for deploying HATS rich client applications. For example, Lotus Expeditor Server can be used to centrally manage the software deployed to Lotus Expeditor Client systems.

3. The user, running a rich client platform, downloads and installs the HATS application feature from the update site.

4. The user starts and accesses the HATS application using the rich client platform environment.

5. The HATS runtime connects to the target host system through a Telnet server.

6. As the user interacts with the host through the HATS application, the HATS runtime transforms host screens to a GUI.

---

HATS rich client application

1. Develop / test

   ![Diagram](image.png)

   - HATS Toolkit
   - Rational SDP
   - Web server
   - Update site
   - Telnet server (3270 or 5250)
   - Rich client

   2. Upload

   3. Download

   4. Access

   5. Connect

   6. Transform
With the HATS Toolkit, you can develop a HATS rich client application that provides an easy-to-use graphical user interface for your 5250 applications running on System i hosts or your 3270 applications running on System z hosts. For example, below is the Main Menu screen displayed by a System i host.

Here is the same Main Menu screen rendered by a HATS rich client application using only HATS default transformation rules.
Using HATS screen customization and transformation functions you can transform host screens to look more like modern applications that conform to your company design for rich client applications. In addition, macros can be used to aid users in navigating the host application. Shown below is the same Main Menu screen that has been transformed to a more modern appearance. Unnecessary data and fields are hidden from view, and data entry aids are provided.

Lab overview

This lab exercise is divided into three parts.

In the first part, Default project, you learn how to develop and test a HATS rich client application that uses the HATS default transformation rules.

In the second part, Template and project settings, you learn how to change the template and project-wide settings for a HATS project, and test your changes.

And, in the third part, Screen customizations, you learn how to create HATS screen customizations, transformations, and macros.
Default project

Create project

1. Click **Start > All Programs > IBM Software Development Platform > Rational HATS 7.1 > HATS Toolkit 7.1**.
2. If you are using a shared workstation in a lab room, on the Select a workspace page, click **Browse**, to browse for and select the following folder, `c:\myhats7lab\myworkspaces`. After you have selected this folder, click **Make New Folder** and give the new folder a unique name (for example, your name). Then click **OK**. If this folder structure does not exist on your lab workstation, ask the instructor what workspace to use.
3. If you are using your own workstation, select whatever folder you want to use as the workspace for this lab exercise.
4. Click **OK**.

Notice the HATS tips that are displayed.

5. To clear the tip, click **OK**.
6. If you want to, select **Do not show any tips** and click **OK**.
7. If you select the **Do not show any tips** option, you can instruct HATS to start showing tips again by navigating to **Window > Preferences > HATS** and selecting the **Show HATS tips** preference.

You will now launch the Create HATS Project wizard.

8. Click the **Create HATS Project** button on the toolbar.

On the HATS Project page

9. For Name: enter **myhats5250rcp** (or whatever name you like).
10. In the Deployment section, select the **Rich client** option and use the defaults. Notice in this case the Target Platform is Eclipse Rich Client Platform v3.2 or v3.3.
11. Click **Next**.
On the Connection Settings page

12. For Host name: enter the name of the System i host. If you are using the IBM demo system on the Internet, enter `iseriesd.demos.ibm.com`. Otherwise, ask your instructor for the name of the System i host to use.
13. For Type: select **5250**
14. Click **Next**

On the Project Theme page

15. Notice the different appearance and behavior themes and settings.
16. Take the default (Standard) appearance and behavior theme.
17. Click **Next.**
On the Default Template page

18. From the list of templates select each template to see the results in the Preview panel. Select the Modern template to use for this lab exercise.

19. Click Finish.

After finishing, the Project Settings Overview is displayed. Click the X on the settings tab to close this view for now. You will go through it later in the lab.

In the HATS Projects view, notice your project including all of its folders. You will be looking into these folders as you go through this lab.

**Test project**

In the HATS Projects view, right-click your project’s folder and select Debug.
20. Click **Yes** to see a view of the host display terminal while testing your application. This helps in debugging.

On the Create, manage, and run configurations page, HATS automatically creates a default launch configuration for you.

21. Notice the name of the default launch configuration is hostaccess.
22. Click **Debug**.
A new instance of the Eclipse workbench is launched as a local test environment, the Host Access perspective is automatically opened, and your HATS rich client application is displayed in the Applications view.

You will now start your HATS rich client application.

23. Right-click your application and select **Open**.

In the transformation view, see your HATS rich client application running live with your host system. This is the HATS default transformation for your host system. As you go through this lab, you will learn how to change some of the project settings that affect the default transformation as well as create custom transformations for several screens.
Notice the template area. In the case of the Modern template, as shown below, it is the sidebar area to the left of the host screen transformation. In the section, Work with the template, on page 19, you will see how to edit the template.

Notice the host screen transformation area.

24. In the upper right corner notice the Default and Refresh keys. This is the application keypad.
25. At the bottom of the host screen, notice that there are no buttons for host keys visible. This is because the host keypad is not displayed by default.
26. Finally, notice the Operator Instruction Area (OIA) that is displayed by default. At this point all you see is the display of the cursor row and column numbers.
27. In the section, Change project settings, on page 26, you will see how to change these global project settings.

In the top right corner, click the twistie icon to display its menu items.

From the menu, select Connection Details and notice the displayed results.
Also, notice the host display terminal that shows the actual host screen that HATS is transforming. The user never sees this screen. It is displayed because you requested it, in an earlier step above, and it can be used to help you debug your HATS application.

Navigate the host application

Next, you will navigate through the host application to see how HATS transforms each of the screens using the default transformation.

On the Sign On screen:

28. Use userid ATDEMO password DEMO4YOU
29. If this userid doesn't work using the Internet demo system, iseriesd.demos.ibm.com, use the userid and password documented in http://websphere.dfw.ibm.com/atdemo/atdemo_uid_pw.pdf. If you are using a host system in a lab room, ask the instructor for the userid and password.

On the Display Program Messages screen:

30. If you see this screen, press the Enter key.

On the Main Menu screen:

31. Notice the items on the toolbar. This is how the function keys, recognized on the host screen, are rendered by default.

32. Looking at the host display terminal, compare the function keys on the host screen below with the toolbar items above.
33. Notice that the list of selections is rendered using buttons. By default, for rich client applications, HATS renders selection lists using a button table.

34. To navigate to the application used for this lab, on the command line type `strqm` and press the Enter key.

On the DB2 Query Manager screen:

35. From the selection list, click button number 3, Work with Query Manager tables.
On the Specify Collection or Library screen:

36. For Library, type **whidemo** and press the **Enter** key.

On the Work with Query Manager Tables screen:

37. In the application keypad area of the toolbar, click the **Default** button to see the basic host screen, then click the **Refresh** button to see HATS current rendering for the screen. Notice how HATS renders the subfile that is recognized.
38. Using the drop-down next to the JK_CATALOG table, select \textbf{5=Display table}.

On the Find Data in Query Manager Table screen:

39. Under Column type \textit{product\_category}.
40. Under Test type \textit{eq}
41. Under Value type \textit{Animals} (with an uppercase A)
42. Press the \textbf{Enter} key.

On the Display Report screen:

43. Press the \textbf{PgDn} key to page through all of the results.
On the last Display Report screen:

44. Click **Exit** on the toolbar (the F3 function key).

On the Work with Query Manager Tables screen:

45. Click **Exit** on the toolbar (the F3 function key).

On the DB2 Query Manager screen:
46. Click **Exit** on the toolbar (the F3 function key).

On the Main Menu screen:

47. Click the button number **90 Sign off**.

Minimize the local test environment and the host display terminal. You will use them again later in the lab.
Template and project settings

Work with the template

HATS rich client templates are specialized Java SWT composites. They are very basic with a simple header or sidebar and no footer. This is because it is assumed that the runtime environment (for example, Eclipse RCP or Lotus Expeditor) will provide for the overall common appearance as well as areas for common buttons and links.

1. If not already there, switch to the HATS Projects view by clicking the HATS Projects tab at the top of the left frame.
2. In the HATS Projects view expand the Rich Client Content Templates folder.
3. Double-click the template you selected as your default. This lab uses the Modern template.

This launches your template in the visual editor for Java. There are several views you will use while editing your template.

4. The Design view previews the Java beans for the components in your template in a graphical representation. You can use the Design view to change some of the properties of the Java beans. Other properties are changed in the Properties view.
5. The Source view shows the associated Java code.
6. The Palette view displays components and other selection tools for you to use in the Design view.
7. The Java Beans view shows a tree view of the components, events, and listeners used by your template.
8. The Properties view shows the properties for the Java bean that is selected in either the Design view or the Java Beans view. You can use the Properties view to change the value of a given property.
9. In this example, you will add some text above the image in the template.
10. In the Java Beans view, with the mouse, select the sideBarArea. You could also select the sideBarArea in the Design view.
11. Notice the selected area in the Design and Source views.
12. From the Palette view, click the **SWT Controls** drawer and select the **Label** component.

13. Next, do not drag, but instead simply position the mouse in the Design view above the image as shown below and click to insert the new label component in a row above the image. Be sure the Design view looks as shown below before you click the mouse.

14. For Bean Name, enter **companyName**, or whatever you want, and click **OK**.
15. Notice the new label in the Design view and the new bean in the Java Beans view.
16. With the new label selected in the Java Beans view, in the Properties view change the properties as shown in the following steps.

   17. Select the **font** property and click the **button** that is displayed at the right margin of the Value column.
18. Set the font Style to **Bold** and click **OK**.

19. Select the **background** property and click the **button** to change the color. On the RGB tab set the RGB values to **221, 228, 255** to match the sideBarArea background. Click **OK**.

20. Change the layoutData, horizontalAlignment property to **CENTER**.
21. Finally, change the text property to **JK Enterprises**, or whatever you want.

22. In the Java Beans view, select the bean for the label you just added.

23. In the Source view, notice the source code for the bean.

   ```java
   companyName = new Label(sideBarArea, SWT.NONE);
   companyName.setText("JK Enterprises");
   companyName.setBackground(new Color(Display.getCurrent(), 221, 228, 255));
   companyName.setLayoutData(gridData);
   companyName.setFont(new Font(Display.getDefault(), "Tahoma", 11, SWT.BOLD));
   ```

24. There is a known Eclipse problem that could result in resource leaks documented in **Bug 100144**: Need a pattern for cleaning up font/image/color resources at [https://bugs.eclipse.org/bugs/show_bug.cgi?id=100144](https://bugs.eclipse.org/bugs/show_bug.cgi?id=100144). Due to this bug, you must add clean-up (dispose) code manually if you add font, image, or color resources to HATS rich client templates or transformations. You do not need to add clean-up logic if you add a system color, which is allocated by the system. For more information, refer to the HATS rich client considerations and limitations section in the HATS V7 User’s and Administrator’s Guide.

25. For this lab exercise, you do not need to worry about adding the clean-up code. The lab exercise will work without it. But for future reference, in case you are creating an application for a production environment, the code to clean-up (dispose) both the color and the font resources for the label you just added is shown in the Source view below.
26. If you want to add the code, you can cut and paste it from below. But is it not necessary to do so for the lab exercise to work.

```java
companyName.addDisposeListener(new DisposeListener() {
    public void widgetDisposed(DisposeEvent e) {
        Color background = companyName.getBackground();
        if (background != null && !background.isDisposed()) {
            background.dispose();
        }
        Font font = companyName.getFont();
        if (font != null && !font.isDisposed()) {
            font.dispose();
        }
    }
});
```

You will now import an image into your project.

27. From menu bar select **File > Import > General > File System** and click **Next** to open the Import wizard.

28. Use the **Browse** button next to the **From directory** field to locate and select an image to import into your project. You can use any image on your system. This lab uses an image that is installed with HATS in the directory, `c:\Program Files\IBM\SDP70Shared\plugins\com.ibm.hats_7.1.0.nnna\predefined\projects\new\Web Content\common\images`. Navigate to and select this directory and click **OK**. If this directory does not exist on your lab system, ask the instructor which directory to use.

29. Select the **MountainPhoto.jpg**.

30. Click the **Browse** button next to the **Into folder** field, expand your project, and select the `\src\images` folder. Click **OK**.
31. Click **Finish**.

Now you will change the image in your template.

32. In the Java Beans view, select the bean named **label**, or in the Design view, select the **image** in the sidebar.

33. In the Properties view, select the **image** property and click the **button** displayed at the right margin of the Value column.

34. In the Select an Image File dialog, under Browse in, select **Project**, under Location expand your **src** folder and click the **images** folder, under Files select the image you imported, for example **MountainPhoto.jpg** and click **OK**.
35. In the Design view, notice the new image in your template.
36. To remove warnings on the Problems tab about unused import statements, select the template source with your mouse, and press Ctrl+Shift+O to organize the import statements.

Close the template editor. Either press Ctrl-s to save your changes first, or close the editor and when prompted click Yes to save the changes.
Note: If you get the dialog, “Hot code replace failed”, click Continue to continue without restarting the local test environment or click Restart to restart the local test environment. In this case you can do either. You will be returning to the local test environment later in this section.

**Change project settings**

37. In the HATS Projects view, double-click Project Settings.

38. In the Project Settings editor click through all of the different tabs at the bottom to see the various things that can be set.
39. When you are through, click the Rendering tab.
40. On the Rendering tab, you can see all of the default rules that HATS uses to recognize and transform screens.
41. Select Default Rendering in the left panel. Notice that for default rendering, selection lists are rendered using the Button table widget and function keys are rendered using the Toolbar widget. You saw the effects of this when you navigated the host system earlier in this lab.
42. Still on the Rendering tab, expand the list of **Components**. These are the components that HATS can recognize on a host screen and the default rules for recognizing them. For example, notice the default rules for how HATS will recognize an area of the screen that represents a function key.

43. When you are finished looking at the various Components, expand the list of **Widgets** that can be used to render a graphical view of a component. Notice the definition for the Toolbar widget. By default any components (for example, function keys) that are rendered with the Toolbar widget are contributed to the main toolbar.
44. When you are finished looking at the various widgets, click the definition for the main **Toolbar**. Notice that by default the toolbar is displayed, and items are displayed as text. Since by default, recognized function keys are rendered using the Toolbar widget, function keys are displayed on the toolbar as text items.

45. Now click **Application Keypad**. Earlier in this lab you saw the Default and Refresh keys displayed on the application keypad. Here you can change what keys to display. Because the Default key is useful for debugging purposes, leave it selected during development. However, you may not want to display the Default key to your users. As an example for this exercise, select the **Disconnect** key so that it will appear on the application keypad.
46. Click **Host Keypad**. Notice that by default the host keypad is not displayed. Here is where you can tell HATS to display the host keypad and what keys to include. For this lab, use the default not to show the host keypad.

47. Next, click **Operator Information Area**. Here is where you can configure what to display in the OIA or choose not to show it at all. For this lab exercise, clear the **Show OIA** check box.

48. Finally, click the **Other** tab and select **Asynchronous Update**. Be sure **Enable asynchronous updates** is selected. This will improve the response time of your HATS application by shortening the length of time HATS waits to be sure a screen has been completely received from the host.
49. Close the **Project Settings View** by clicking the X and click **Yes** to save your changes.

![Project Settings View](image)

Configure default rendering, global rules, text replacement, and

**Test your changes**

50. Switch back to your application running in the local test environment, and click the **Refresh** key on the application keypad.

51. If you get the Disconnected screen, click **Restart**.

52. If you got the “Hot code replace failed” dialog while saving your template, and you clicked **Continue**, you must exit the local test environment and start your application again to see the changes. To exit, from the menu bar select **File > Exit**. Then start your application again from the HATS Projects view by right-clicking your project and selecting **Debug**.

![Host Access](image)

53. After you have either refreshed or restarted your application, notice the Disconnect key is displayed on the application keypad, the OIA is no longer shown, and the changes you made to your template are applied.
54. Minimize or exit the local test environment.

Screen customizations

In the following sections are exercises for customizing screens from the host application. HATS is very flexible with the method you use to customize screens.

1. One method is to capture all of the screens you want to customize and create all of the macros you want to use first, then go back and customize your screens.
2. Another method is to capture a screen, record any macros necessary for the screen, and then customize the screen before going on to the next screen.
3. In this lab, you will use the second method so that you can have a more self-contained exercise for each screen customization.

If you are running short on time, and only want to customize one screen at this time, then go directly to the instructions for customizing the Main Menu screen on page 49. You can finish the remainder of the lab at home using an evaluation copy of HATS V7 and the demo system (iseriesd.demos.ibm.com) on the Internet.

Sign On screen

Create screen capture

In the HATS Projects view, right-click your project and select Open HATS Host Terminal > Main

Here you see the Sign On screen from the host system.

4. On the toolbar click the Create Screen Capture button.

5. On the Capture a Screen page, accept the default screen capture name, SignOn, and click Finish.
Record macro

HATS macros can be played either by clicking a button or automatically when a screen from the host is recognized. In this case, you will record a macro that will sign on with a generic userid. Of course, if the application requires a user to sign on with a unique userid, you would not create this signon macro. But this example uses a shared generic userid that has been set up to allow read only access to just a subset of the applications and transactions on the host.

Before recording your macro, use the local test environment to sign on to the host using the lab userid. This will force the appearance of the Display Program Messages screen as you record your macro. Later in the lab, you will see how to edit your macro to handle the case when the Display Program Messages screen does not appear.

For the iseriesd system use the same userid ATDEMO password DEMO4YOU that you used before.

If you are using a host system in a lab room, use that userid and password, the same as before.

To start recording the macro, click **Record new macro ...** on the toolbar

On the Record Macro page,

6. Accept the default name for the macro, **SignOn**, and click **Finish**.
The first task in creating a macro is to tell HATS how to recognize the screen on which the macro starts. On the Define the starting screen of the macro page:

7. For Screen Name, type **SignOn**.
8. At the bottom, for **Highlight fields**, select at least the **Input** option.
9. Select **Within a rectangular region**.
10. Click **Finish**.

On the Sign On screen, type the userid and password and either press the **Enter** key or click the **Enter** button. Use the same userid and password you used in the local test environment.

On the Display Program Messages screen, click **Define Screen Recognition Criteria**.
11. For Screen Name, type **DPM**.
12. Drag your mouse around just the screen title, **Display Program Messages**, at the top. Be sure not to include any dates or times in your screen recognition criteria because when the date or time changes, the screen will no longer be recognized.
13. Select **At a specified position**.
14. Click **Finish**.

On the Display Program Messages screen, either press the **Enter** key or click the **Enter** button.

On the Main Menu screen, from the toolbar click **Stop Macro**.
15. For Screen Name, type **MainMenu**.
16. With your mouse draw a box around the text, **MAIN**, at the top of the screen.
17. Select **At a specified position**.
18. Click **Finish**.

Next, edit the macro to handle the case when the Display Programs Messages screen does not appear, that is, the first time the lab userid is used.

In the Macro Navigator panel, right-click the **Next Screens** folder following the SignOn screen and select **Edit**.
Under Available Screens, select the entry for the MainMenu screen and click the button to move it under Next Screens. After doing this, the panel should look as shown below, Click OK.

Notice in the Macro Navigator after the SignOn screen, the Next Screens entry will look for either the DPM screen or the MainMenu screen and branch to execute accordingly. Click Save Macro.

To test your macro, navigate back to the Sign On screen in the host terminal window. Do this either by typing 90 in the ‘Selection or command’ field on the host screen and pressing Enter, or by disconnecting and reconnecting the host session from the buttons on the host terminal toolbar.
After you are back to the Sign On screen, from the host terminal toolbar select the **Play Macro dropdown** and select the **SignOn** macro.

Now you could continue to capture more screens, and after capturing all the screens you want, then start customizing them. However, these exercises are written so that the work for each screen can be in a self-contained exercise. So after your macro is saved, click the **Disconnect** button on the toolbar to disconnect the terminal session and close the terminal window.

Create screen customization

You have created a screen capture for the SignOn screen and a SignOn macro. Next you will create a screen customization for this screen. A screen customization consists of how to recognize the screen and what actions to perform once the screen is recognized.

Notice in the Screen Captures folder there is an entry for the SignOn screen, and in the Macros folder there is a SignOn macro. Double-click the **SignOn** screen capture.

Here you see a rendering of the SignOn screen capture. Click the **Preview** tab at the bottom.
Here you see how HATS will transform this screen based on the current default rendering rules in the project settings.

Click the **Host Screen** tab and select **New Screen Customization** from the drop-down at the top.

Accept the supplied screen customization name of **SignOn** and click **Next**.
The first task in creating a screen customization is to tell HATS how to recognize the screen. On the Screen Recognition Criteria page, accept the default recognition criteria and click Next.

After you tell HATS how to recognize the screen, then you tell HATS what to do once the screen is recognized. On the Actions page, you see that you can Apply a transformation, which means to display the screen to the end user in a transformed way. Also, you can tell HATS to play a macro or perform advanced actions when this screen is recognized. In this case, just select Apply a transformation and click Finish.
Create screen transformation

You are back in the workbench view. Notice in the HATS Projects view there are entries for the SignOn screen in both the Screen Customizations and the Rich Client Content\Transformations folders.

HATS rich client transformations are Java SWT composites and by default are edited using the visual editor for Java.

Your new transformation has been launched in the visual editor for Java. There are several views you will use while editing your transformation. For a description of these views see the section, Work with the template.

You will now add text to provide a screen or system name for your transformation.

19. From the Palette view, click the SWT Controls drawer and select the Label component.
20. Next, do not drag, but instead simply position the mouse in the upper left corner of the Design view, as shown below, and click to insert the new label component. Be sure the Design view looks as shown below before you click the mouse.

21. Give this bean a name, for example, `screenName`, and click OK.

22. Notice the bean for the component you just added in the Design view.
23. In the Java Beans view.

24. In the Source view.

```java
private void initialize()
{
    screenName = new Label(this, SWT.NONE);
    screenName.setText("Label");
    GridLayout layout = new GridLayout(1, false);
    setLayout(layout);
    this.setSize(new org.eclipse.swt.graphics.Point(500, 300));
}
```

25. And, in the Properties view.

26. You can use the Properties view to change any properties for your bean. You can also change some of the properties by right-clicking the bean in either the Design or the Java Beans view. In this case, right-click the bean in the Design view and select **Set Text**.
27. Enter the text you want to appear on your transformation. In this case enter, **JK Products**, and press **Enter**.

![Image of a label with "JK Products" text]

28. Using the Properties view you can change other properties. For example, to use a bold font, in the Properties view select the **font** property and then click the **button** at the right margin of the Value column.

![Image of a Properties view with "font" property selected and "Tahoma, 11" as value]

29. Set the style to **Bold** and click **OK**.

![Image of a Properties view with "text" property selected and "JK Products" as value]
30. You can use the Properties view to change the layout data properties, for example horizontal alignment, horizontal span, vertical alignment and vertical span, to further position your beans on the transformation. You can experiment with changing these other properties. But for now, the lab will continue with creating the transformation.

Add a second line of text to provide instructions to the user.

31. From the Palette view, select **Label** in the SWT Controls drawer.
32. Position the mouse in the Design view below the first text you added, as shown below, and click to drop.

33. Give this bean a name, for example, **instructions**, and click **OK**.

34. Again, notice the newly added bean in all of the different views.
35. As before, right-click the bean in the Design view, select **Set Text**, and add instructional text for the user. For this lab enter, **Click below to sign on**. Tip: if you want to create a blank line, instead of typing new text here, simply delete the original text supplied with the bean.
Add a HATS Macro Key to run the SignOn macro when clicked.

36. From the Palette view, select **Macro Key** in the HATS drawer.

37. Position the mouse in the Design view below the second line of text you added, as shown below, and click to drop.

38. In the Insert Macro dialog, select which macro you want to play, in this case there is only one. Notice that you can select to display the macro key as either a button or a link. Select **Button** and click **OK**.
39. Notice the Design view shown below.

40. Also notice the Java Bean view, shown below, and the other views.

Note: See the information about eclipseBug100144 in step Error! Reference source not found. and the need to add clean-up code for font, image, and color resources to avoid resource leaks. It is not necessary to add the clean-up code for this lab exercise to work. But for future reference, in case you are creating an application for a production environment, the code to clean-up (dispose) the font resource you changed in this transformation is shown in the Source view below.
To remove warnings on the Problems tab about unused import statements, select the transformation source with your mouse, and press Ctrl+Shift+O to organize the import statements.

You have finished creating a screen customization and screen transformation for the Sign On screen. Exit and save the screen transformation by closing the tab for the SignOn.java editor and clicking Yes to save changes.

From the screen capture view, click the Preview tab.
Here you see a preview of how HATS will transform this screen using the transformation you just created.

Close the screen capture view by clicking the X.

**Test screen customization and transformation**

Switch back to the local test environment and click the **Refresh** button on the application keypad.

Notice your transformed Sign On screen (**JK Products**).

Restore the host display terminal window if you want to see the macro play through the host screens.
Click the **SignOn** button to play the SignOn macro.

The SignOn macro plays, signs on using the supplied userid and password, recognizes whether the Display Program Messages is displayed, and navigates to the Main Menu screen.

**Main Menu screen**

**Create screen capture**

In the HATS Projects view, right-click your project and select **Open HATS Host Terminal > Main**

Navigate to the Main Menu screen as described in the section, **Navigate the host application**, on page 12. If you have already created a SignOn macro, then on the toolbar, click the **drop-down** next to the Play Macro button and select the **SignOn** macro. Or, you can simply sign on manually.
After navigating to the Main Menu screen, you will create a screen capture.

41. Click the **Create Screen Capture** button.

42. On the Capture a Screen page, name this screen capture, **MainMenu**, and click **Finish**.

After creating the screen capture, click the **Disconnect** button on the toolbar to disconnect the terminal session and close the terminal window.

**Create screen customization**

You have created a screen capture for the Main Menu screen. Next you will create a screen customization for this screen. A screen customization consists of how to recognize the screen and what actions to perform once the screen is recognized.
Notice in the Screen Captures folder you have an entry for the MainMenu screen. Double-click the MainMenu screen capture.

Here you see a rendering of the Main Menu screen. Click the Preview tab at the bottom.

Here you see how HATS will transform this screen based on the current default rendering rules in the project settings.
Click the **Host Screen** tab again and select **New Screen Customization** from the drop-down at the top.

Accept the supplied screen customization name of **MainMenu** and click **Next**.

The first task in creating a screen customization is to tell HATS how to recognize the screen. On the **Screen Recognition Criteria** page, with your mouse, draw a box around the text, **MAIN**, at the top left corner of the screen. Click **Next**.

After you tell HATS how to recognize the screen, then you tell HATS what to do once the screen is recognized. On the **Actions** page, notice that you can **Apply a transformation**, which means to display the screen to the user in a transformed way. Also, you can tell HATS to play a macro or perform advanced actions when this screen is recognized. In this case, just select **Apply a transformation** and click **Finish**.
Create a Screen Customization - MainMenu

Actions
These actions will be performed when a host screen matches the recognition of customization. You can also add or modify actions later using the screen customizer.

- Apply a transformation
  - Use existing transformation
  - Create new transformation myhats525rcp.transformations.MainMenu

Select a pattern:
- Blank
- Default Rendering
- Prepapulated using Default Rendering
- Prepapulated using fields

Create a screen transformation

You are back in the workbench view. Notice in the HATS Projects view there are entries for the MainMenu screen in both the Screen Customizations and the Rich Client Content\Transformations folders.

HATS rich client transformations are Java SWT composites and by default are edited using the visual editor for Java.

Your new transformation has been launched in the visual editor for Java. There are several views you will use while editing your transformation. For a description of these views see the section, Work with the template.

Add text to provide a screen or system name for your transformation.

43. From the Palette view, select Label in the SWT Controls drawer.
44. Position the mouse in the upper left corner of the Design view, as shown below, and click to insert the label. Be sure the Design view looks as shown below before you click the mouse.

![Diagram showing mouse position in Design view]

45. Give this bean a name, for example, `screenName`, and click `OK`.

![Dialog box showing bean name]  
**Bean Name**

Provide names for beans

- `label="Label"` `screenName`

46. Notice the bean for the component you just added in the different views.

47. Right-click the bean in the Design view and select `SetText`.
48. Enter the text you want to appear on your transformation. In this case enter, **Application Menu**, and press **Enter**.

49. In the Properties view, select the **font** property and then click the **button** at the right margin of the Value column.

50. Set the style to **Bold** and click **OK**.
51. You can use the Properties view to change the layout data properties, for example horizontal alignment, horizontal span, vertical alignment and vertical span, to further position your beans on the transformation. You can experiment with changing these other properties. But for now, the lab will continue with creating the transformation.

Add a second line of text to provide instructions to the user.

52. From the Palette view, select **Label** in the SWT Controls drawer.
53. Position the mouse in the Design view, as shown below, and click to drop.

54. Give this bean a name, for example, **instructions**, and click **OK**.

55. Right-click the bean in the Design view, select **Set Text**, and add instructional text for the user. For this lab enter, **Select application and click Enter**. Tip: if you want to create a blank line, instead of typing new text here, simply delete the original text supplied with the bean.
You will now add a HATS component to prompt the user with a list of applications to choose from.

56. From the Palette view, select **Component** in the HATS drawer.

![Palette view with Component selected]

57. Position the mouse in the Design view, as shown below, and click to drop.

![Design view with application menu selected]

58. The Insert Host Component wizard is launched. On the Screen Region page, in the Highlight fields section, select **Input**. Then use the mouse to select the command line region as shown below. Click **Next**.

![Screen Region page with Input selected]
59. On the Rendering Options page, select the **Input field** component and click the **Component Settings** button to change the default settings for this instance of the input field component.

60. On the Settings page, clear the **Use project defaults** and the **Extract field caption** check boxes. Click **OK**.
61. On the Rendering Options page, select the **Drop-down (data entry)** widget and click the **Widget settings** button to change the default settings for this instance of the drop-down (data entry) widget.

62. On the Settings page, clear the **Use project defaults** check box. Select **Fill from string** and in the **List items** box type **JK Boats=boats; JK Toys=strqm; Sign Off=90;** where for example if JK Toys is selected from the list, then the string, strqm, is passed to the host. The string, strqm, is the command to start the query manager for the JK product tables. If JK Boats is selected, the string, boats, is passed to the host to start the boats application. Click **OK**.
63. Preview your widget in the Widget Preview area, and click **Finish**.

![Widget Preview](image)

64. Notice the drop-down component in the Design view.

![Application Menu](image)

65. To modify the HATS component or widget settings, in the Properties view, select **component** and click the **button** at the right margin of the Value column as shown below. For this lab, no more changes are required.

![Properties View](image)

Add a HATS host key to perform the Enter key function.

66. From the Palette view, select **Host Key** in the HATS drawer.
67. Position the mouse in the Design view to the right of the drop-down component you added, as shown below, and click to drop.

![Diagram showing mouse position and drop-down component]

68. Select the **Enter** key function and click **OK**.

![Diagram showing Insert Host Key dialog]

69. In the Design view, notice the component for the Enter key button.

![Diagram showing Application Menu with Enter button]

Reposition the rows of text to span both HATS components in the third row.
70. In the Design view, select the text in row 1 and with the mouse grab the handle of the cell as shown below.

71. Using the mouse, stretch the cell to span both columns.

72. Release the mouse and notice that row 1 now spans both columns.

73. Do the same procedure for the text in row 2 and notice that both rows of text span both columns.
Note: See the information about `eclipseBug100144` in step *Error! Reference source not found.* and the need to add clean-up code for font, image, and color resources to avoid resource leaks. It is not necessary to add the clean-up code for this lab exercise to work. But for future reference, in case you are creating an application for a production environment, the code to clean-up (dispose) the font resource you changed in this transformation is shown in the Source view below.

```java
screenName.setFont(new Font(Display.getDefault(), "Tahoma", 11, SWT.BOLD));
screenName.addDisposeListener(new DisposeListener()
{
    public void widgetDisposed(DisposeEvent e)
    {
        Font font = screenName.getFont();
        if (font != null && !font.isDisposed())
        {
            font.dispose();
        }
    }
});
```

To remove warnings on the Problems tab about unused import statements, select the transformation source with your mouse, and press `Ctrl+Shift+O` to organize the import statements.

You have finished creating a screen customization and screen transformation for the Main Menu screen. Exit and save the screen transformation by closing the tab for the MainMenu.java editor and clicking **Yes** to save changes.
From the screen capture view, click the **Preview** tab.

Here you see a preview of how HATS will transform this screen using the transformation you just created.

Close the screen capture view by clicking the **X**.
Test screen customization and transformation

Switch back to the local test environment and click the **Refresh** button on the application keypad.

If you get the Disconnected screen, click **Restart**.

Navigate to your transformed Main Menu screen (**Application Menu**).

Restore the host display terminal window if you want to see the interaction with the host when you click the Enter button.

From the drop-down, select **JK Toys** and click the **Enter** button.

The string, strqm, is entered into the command line field, and the Enter key is entered. Next you see the default transformation for the DB2 Query Manager screen.