Handling Screens with Tables/Complex Data

Host tabular data can be easily displayed as an HTML table on the web page. Once a table is defined on the host screen, it will be automatically generated to the web page along with the rest of the screen.

In this exercise you will:

- Add a Screen Based Table (via the Screen Editor)
- Manually add a Table to a JSP/ASPX Page
- Create a Procedure Based Table (from Multiple Screens)
- Export a Table to Excel
- Things to Take into Consideration when Collecting Data from Multiple Screens

---

Add a Screen Based Table (via the Screen Editor)

Refer to Tables in the Basic Exercises section.

 Accompanying movies:

- Creating a Screen Based Table

Manually add a Table to a JSP/ASPX Page

Exercise objectives

Manually create a screen based table without having to generate the screen from scratch. This table can then be used to display a screen based table or can be filled with a data structure array collected with an ApplinX path procedure.

 **Exercise**

Manually add a table to the relevant JSP/ASPX page.

If you already have a web page and don’t want to regenerate it, you can manually add an ApplinX table to a web page by just following these simple rules:

- The table ID must match (case sensitive) the table name in the screen Editor.
- Cells in the table that are to contain values from the host screen must either, have an ID or contain an HTML control that has an ID that matches the column (multiple field) name.
- Usually, a table will hold two rows. The first, a header row. And the second, a row that represents the rest of the data rows in the table.

Refer to BrowseCustomers or BrowseProposals in the Composite demo application.

Create a Procedure Based Table (from Multiple Screens)

Procedure based tables are Web page tables that are bound to a path procedure output data, usually collected from several host screens. For example, in a host screen with a table, clicking a host key will scroll down the table.

Exercise objectives

To “collect” the whole table and display it on one web page, as oppose to having the user click a button to get the next set of table records. The following exercise demonstrates how to collect a table from several screens. This isn’t the only way of doing it, but for this common scenario, of a scrollable table, it is the easiest.

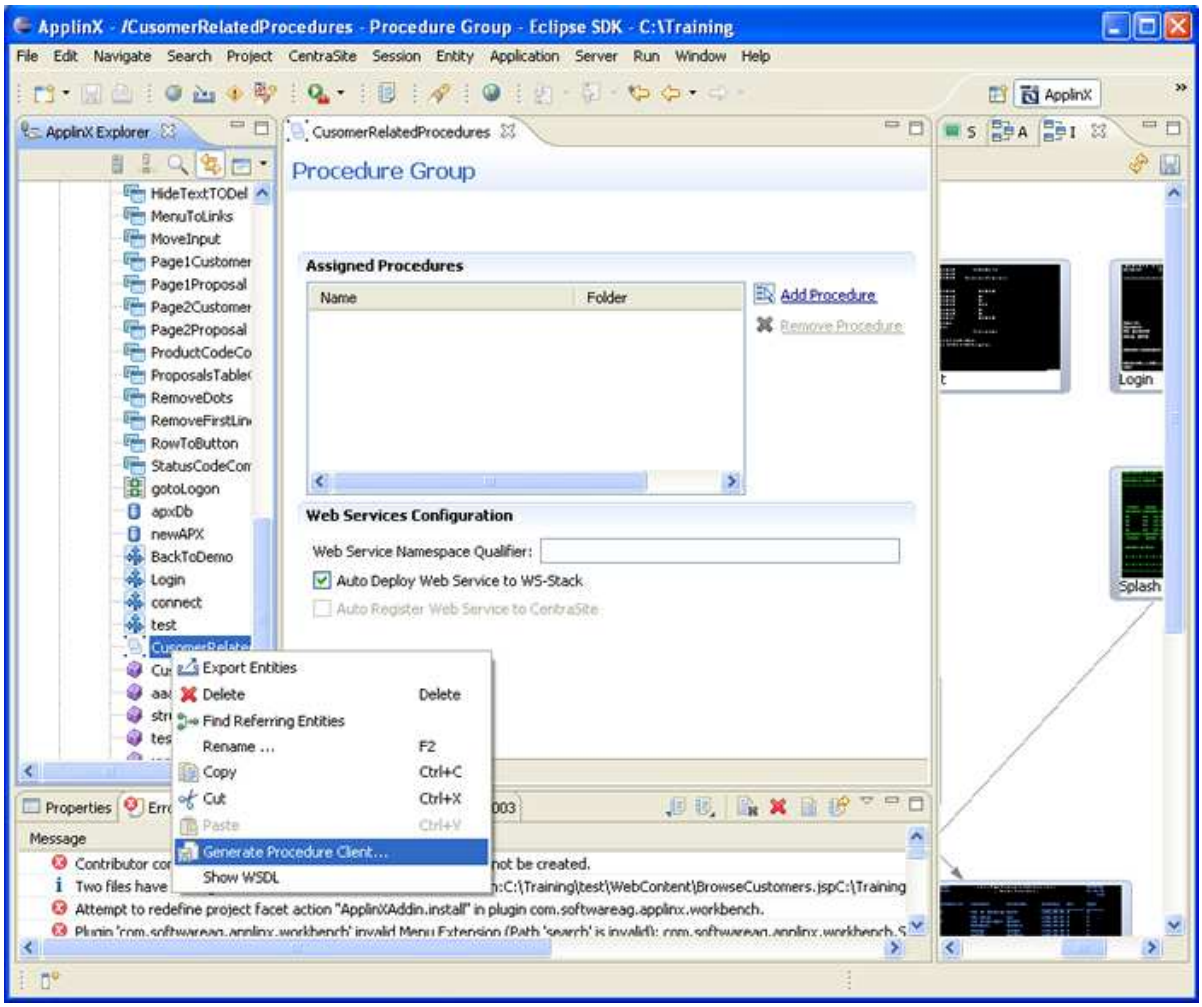
 **Exercise**

Create a path procedure which collects all customer information as output variables and displays the output on the first customer detail web page, preferably, using tabs.

 **Solution steps**

1. Navigate to a screen with table formatted data (BrowseCustomers, Browseproposals).
2. Identify the screen if has not yet been identified.
3. Map all fields, single fields as well as multiple fields which represent the table’s columns.
4. In the Table tab, create a new table, delete unnecessary columns and change column captions to be more user-friendly.
5. Select Primary key columns. These are typically columns with unique values.
6. Record a Path Procedure that scrolls through the table until it reaches an “End of Data” (or something similar) message on the host screen.
7. Open the path procedure for editing: click the step where the path is scrolled and open its output tab.
8. Select all the required fields from the table entity (hold CTRL and click for multiple select). Then right click the selection and click “New Data Structure”. A dialog box is displayed. Enter a name for the new data structure and click OK. You have just created a Structure that represents one row in the table.
9. Click the “Flow Path” node and open the output tab. Click Add structure and name the variable. Select the newly created data structure from the “type” Combo. Check the “array” checkbox, to make your variable actually represent a table (collection of rows).
10. Add a screen mapper node after the “scroll down” step. Select the relevant screen and then drag the table node onto the newly added variable.

11. Finally, associate the procedure with a procedure group. Create a new procedure group in the repository then add the new procedure to the “Assigned Procedures” box by:
    - Clicking the “Add Procedure” button and selecting the procedure from the dialog.
    - Dragging the procedure from the repository on to the “Assigned Procedures” box.
12. Create a table in the JSP/ASPX page. Follow the rules from the previous section with the exception that Table ID must match the output array parameter and cell IDs must match the Data structure properties names.
13. In order to be able to bind the data from the procedure’s output to our JSP/ASPX table, generate a procedure client from the ApplinX application to the JSP/ASPX project.



14. Uncomment the code in the gx_fillForm method and customize it according to your procedure specifications.

Note:
This process can be applied not only for binding table data, but also to fill other elements of the web form. For example, if the procedure response has an output variable named “message” and the JSP/ASPX page contains a SPAN tag Named(ID) “message” as well, that span will also be bound to the response data. This way multiple host screens can be displayed on a single web page.

 Accompanying movies:

- Collecting Complex Data from Several Screens
- Generating a Procedure Client (JSP)
- Generating a Procedure Client (.NET)

Export a Table to Excel

ApplinX data can be easily exported to an MS Office application, such as an Excel spreadsheet. It is possible to modify the styling and formatting that will be displayed in Excel. This modification can be done in the HTML tags of the XSL file (For example, if you want to add a border to a table, in the excel.xsl file, add to the table tag: <table border=1>).

 **Exercise**

Export a table from a screen which uses a table.

 **Solution steps**

Once we have a GXTable object we can export it to Excel by filling an HTML table:

- The page response content type is set to Excel.
- The page inherits from GXBasicContext/GXBasicWebform. Refer to the Composite demo, Excel page, used in both browseCustomers and browseProposals.

 Recommended reading in ApplinX documentation:

- Web Application Development > Page Customization > Exporting an Data to MS Office (Excel, Word)

Things to Take into Consideration when Collecting Data from Multiple Screens

Amount of Data

As we've seen before, it is possible to merge multiple host screens into a single view on the web page, but when should we do this? Here are a few issues we should take into consideration:

- Do the screens have the same context? For example, it is logical to merge customer details spread across a number of host screens into a single web page.
- How many screens are there?

The more host screens you go through, the longer the end user has to wait. Don't try collecting a host table with thousands of rows. Instead, you can let the users use the scroll functionality the host offers or collect 4-5 screens each time the users scrolls.

- What kind of functionality does the host offer in each of these screens, and can it be duplicated/omitted when all screens are merged in to one?

Validating the Web Page Contents

When you merge several host screens into one web page, using JavaScript you can try and perform some of the host's input validations on the client side before the data is submitted to the host or even while the user is typing. Other validations can be performed on the web server side before the data is sent to the host. This is recommended in order to prevent the host.

- Check for required fields.
- Validate date strings, add a calendar Icon and bind it to the date input.
- Check String formats (number, alphanumeric, email, phone etc.)
- Some interaction between input fields.

 Refer to Combining Data from Multiple Host Screens in the ApplinX Web Application Development Guide>Working with the Framework section of the ApplinX documentation.