Generating and Customizing Screens

Exercise Objectives

Now that we have created a new project/web application to work with, we will customize host screens. In this exercise we will customize the *Login* screen. When you have completed this exercise, you will know how to:

- Generate a Screen
- Record a Path Procedure
- Add a Button which Executes the Path Procedure
- Login Page

Generate a Screen

 **Exercise**

1. Generate the Login screen.
2. Open the JSP file and remove unnecessary data and inputs. Apply your own design to the page and leave only essential fields on the screen (username, password, reconnect and message).

 Accompanying movies:

- Generating a screen (JSP)
- Generating a screen (.NET)

Record a Path Procedure

When reviewing the logic of the login process, notice that the Splash screen and Menu screen have no substantial use and the user should not have to pass through these screens.

The logical behavior is that the user will go from the Login screen, straight to the InsuranceMenu screen. To implement this, you must create a Path Procedure.

Exercise

1. Record a Login path that navigates from the Login screen to the InsuranceMenu screen:
   - Ensure that the path procedure has at least two input variables - user name and password.
   - Ensure that the path procedure has an output variable - errMsg.
2. After recording the path procedure, edit it so that is also handles situations where the wrong user name or password were entered.

 Accompanying movies:

- Recording a Path Procedure

Add a Button which Executes the Path Procedure

The login path procedure can be executed by clicking on a button.

Exercise

Create a button that will execute the path procedure with values from the web page and will navigate straight to the InsuranceMenu screen.

Solution steps

1. Add the following to your web page, wherever you see fit:

   JSP

   ```
   <gx:input type=”button” value=”go!!” id=”LoginButton” onserverclick=”doLogin” />
   ```

   .NET

   ```
   <input type=”button” value=”go!!” runat=”server” id=”LoginButton” onserverclick=”doLogin”/>
   ```

2. Add the following code to you code behind class (java / C#):

   Java

   ```
   public void doLogin(){
     try {
           GXPathRequest req = new GXPathRequest("Login");
                                          String _user =  getTagsAccesor().getTagContent("UserID");
           String _pwd = getTagsAccesor().getTagContent("Password");
           req.addVariable("UserID", _user);
           req.addVariable("Password", _pwd);
               GXPathResponse res = getGXSession().executePath(req);
           gx_handleHostResponse();
     } catch (GXGeneralException e) {
           gx_handleSessionError(e);
     }
   }
   ```

.NET

```
public void doLogin(object sender, EventArgs e)
{
   GXPathRequest req = new GXPathRequest("Login");
   req.addVariable("UserID", UserID.Value);
   req.addVariable("Password", Password.Value);
   GXPathResponse res = gx_session.executePath(req);
   gx_handleHostResponse();
}
```

Login Page

In order to display the Login web page as the first page of the web application, without having an actual ApplinX session started, we'll need to make it stand-alone and not dependent on the ApplinX session.

Exercise objectives

The scenario we're interested in is: the Login page comes up, regardless of the ApplinX session. The user enters a user name and password and then the Login page creates a new ApplinX session and executes the Login path. The following steps detail how this is done:

Exercise

1. Change the default inheritance of the Login.java (or cs/vb) class from GXDefaultLogicContext (.NET: GXDefaultLogicWebForm) to GXBasicContext (.NET: GXBasicWebForm). This prevents the page from automatically trying to attach itself to an existing ApplinX session. However, it also prevents the page from running the gx_fillForm method which fills all the form's inputs and labels with host data. We'll have to handle that ourselves.

2. If the host application has other screens before the actual login screen, add these screens to the Login path.

3. Change the doLogin method to handle creating a new ApplinX Session or attaching to an existing one.

   JSP

```
public void doLogin(){
  try {
        String _user = getTagsAccesor().getTagContent("UserID");
        String _pwd = getTagsAccesor().getTagContent("Password");

                   GXIConfig config = getGXAppConfig().get_SessionConfig();
                   config.setSessionId(_user);
                   config.setPassword(_pwd);
                // gx_connect() will attach the Page to an existing ApplinX
                // session that has the provided ID. if none exists
                // a new session will be created.
                gx_connect();

                GXPathRequest req = new GXPathRequest("Login");
        req.addVariable("UserID", _user);
        req.addVariable("Password", _pwd);
                GXPathResponse res = getGXSession().executePath(req);
```

```
                gx_handleHostResponse();

   }
   catch (GXGeneralException e) {
                gx_handleSessionError(e);
   }
}
```

.NET

```
public void doLogin(object sender, EventArgs e)
{
        string _user = UserID.Value;
        string _pwd = Password.Value;

        GXIConfig config = gx_appConfig().SessionConfig;
        config.setSessionId(_user);
        config.setPassword(_pwd);
        // gx_connect() will attach the Page to an existing ApplinX
        // session that has the provided ID. if none exists
        // a new session will be created.
        gx_connect();


   GXPathRequest req = new GXPathRequest("Login");
   req.addVariable("UserID", _user);
   req.addVariable("Password", _pwd);
   GXPathResponse res = gx_session.executePath(req);
   gx_handleHostResponse();
}
```