Creating the Natural Code

This chapter contains the following exercises:

- Importing the Adapter into Natural
- Creating the Main Program
- Testing the Completed Application

Importing the Adapter into Natural

You will now import the generated adapter into Natural to make it available to your application.

When you saved your page layout, Application Designer created the Natural adapter `HELLO-A` for your page. This is the name that you have specified earlier in this tutorial. Your application program will use the adapter to communicate with the page. The adapter has been generated into the following directory:

<installdir>/cisnatfirst/nat

Note:
The location of *<installdir>* depends on your application server environment.

To import the adapter

1. Import the adapter source into the Natural library `CISHELLO` which you have created earlier in this tutorial. To do so, use either drag-and-drop or the import function of the `SYSMAIN` utility.

   The adapter code looks as follows:

```
* PAGE1: PROTOTYPE       --- CREATED BY Application Designer --- /*<RO>>
* PROCESS PAGE USING 'XXXXXXXX' WITH
* NAME RESULT
DEFINE DATA PARAMETER
1 NAME (U) DYNAMIC
1 RESULT (U) DYNAMIC
END-DEFINE
*
PROCESS PAGE U'/cisnatfirst/helloworld' WITH
PARAMETERS
 NAME U'name'
  VALUE NAME
 NAME U'result'
  VALUE RESULT
END-PARAMETERS
*
*  TODO: Copy to your calling program and implement.
/*/*( DEFINE EVENT HANDLER
* DECIDE ON FIRST *PAGE-EVENT
*  VALUE U'nat:page.end'
*   /* Page closed.
*   IGNORE
*  VALUE U'sayHello'
*   /* TODO: Implement event code.
*   PROCESS PAGE UPDATE FULL
```

```
*  NONE VALUE
*   /* Unhandled events.
*   PROCESS PAGE UPDATE
* END-DECIDE
/*/*) END-HANDLER
*
END /*<<RO>
```

2. Stow the adapter.

Creating the Main Program

You will now create the main program which uses the adapter to display the page and which handles its events. The name of the program will be `HELLO-P` and you will store it in the library `CISHELLO`.

This description assumes that you are working with Natural Studio.

To create the main program

1. Make sure that the library `CISHELLO` is selected.
2. From the **Object** menu, choose **New** > **Program**.
3. Enter a `DEFINE DATA` statement:

   ```
   DEFINE DATA LOCAL
   END-DEFINE
   ```

4. Import the adapter interface into the `DEFINE DATA` statement:
   1. Place the cursor in `END-DEFINE`.
   2. From the **Program** menu, choose **Import**.
   3. In the resulting dialog box, select the **Adapter** option button.
   4. Select the object `HELLO-A`.
   5. Select all importable data fields.
   6. Choose the **Import** button.

   The result is your completed `DEFINE DATA` statement:

   ```
   DEFINE DATA LOCAL
   1 NAME (A) DYNAMIC
   1 RESULT (A) DYNAMIC
   END-DEFINE
   ```

5. Enter the `PROCESS PAGE` statement. The statement uses the page adapter to display the page in the web browser and to pass data to the controls on the page:

```
DEFINE DATA LOCAL
1 NAME (A) DYNAMIC
1 RESULT (A) DYNAMIC
END-DEFINE
*
PROCESS PAGE USING 'HELLO-A'
WITH NAME RESULT
```

6. Initialize the page data. In the page layout definition, the property `name` has been bound to the FIELD control with the label **Your Name**. For the property `name`, a parameter `NAME` has been generated into the parameter data area of the adapter. Thus, in order to preset the FIELD control, we will preset the variable `NAME` with the value "Application Designer".

```
DEFINE DATA LOCAL
1 NAME (A) DYNAMIC
1 RESULT (A) DYNAMIC
END-DEFINE
*
NAME := 'Application Designer'
PROCESS PAGE USING 'HELLO-A'
WITH NAME RESULT
```

7. Handle the events that can occur on the page. A template for the event handler code has been generated as a comment block into the page adapter `HELLO-A`. List the adapter `HELLO-A` and copy this comment block into your main program and terminate the program with an `END` statement:

```
DEFINE DATA LOCAL
1 NAME (A) DYNAMIC
1 RESULT (A) DYNAMIC
END-DEFINE
*
NAME := 'Application Designer'
PROCESS PAGE USING 'HELLO-A'
WITH NAME RESULT
*
DECIDE ON FIRST *PAGE-EVENT
  VALUE 'nat:page.end'
  /* Page closed.
    IGNORE
  VALUE 'sayHello'
  /* TODO: Implement event code.
    PROCESS PAGE UPDATE FULL
  NONE VALUE
  /* Unhandled events.
    PROCESS PAGE UPDATE
END-DECIDE
*
END
```

After the page has been displayed, the user raises events on the page by using the controls. The name of the raised event is then contained in the system variable `*PAGE-EVENT`. Depending on the event, the program modifies the page data, resends it to browser with a `PROCESS PAGE UPDATE FULL` statement and waits for the next event to occur.

The predefined event `nat:page.end` is raised when the user closes the page. The event `sayHello` is raised when the user chooses the **Say Hello** button. Previously in this tutorial, you have bound the event `sayHello` to this button while designing the page. The `NONE VALUE` block should always be defined as above. It contains the default handling of all events that are not handled explicitly.

8. When the event `sayHello` occurs, we want to display a greeting in the FIELD control with the label **Result**. Therefore, we modify the variable `RESULT` (which is bound to the corresponding FIELD control in the page layout) accordingly before we resend the page data.

```
DEFINE DATA LOCAL
1 NAME (A) DYNAMIC
1 RESULT (A) DYNAMIC
END-DEFINE
*
NAME := 'Application Designer'
PROCESS PAGE USING 'HELLO-A'
WITH NAME RESULT
*
DECIDE ON FIRST *PAGE-EVENT
  VALUE 'nat:page.end'
  /* Page closed.
    IGNORE
  VALUE 'sayHello'
  /* TODO: Implement event code.
    COMPRESS 'Hello, ' NAME '!' TO RESULT
    PROCESS PAGE UPDATE FULL
  NONE VALUE
  /* Unhandled events.
    PROCESS PAGE UPDATE
END-DECIDE
*
END
```

   The main program is now complete.

   If you have not yet saved the program, save or stow it now with the name "HELLO-P".

9. Catalog all modules in the library `CISHELLO`.

Testing the Completed Application

You will now run the application in your web browser and check whether it provides the desired result.

The generated HTML file *helloworld.html* (which is updated each time you save your layout) can be found within the root of your application project, that is in *<installdir>/cisnatfirst*.

This HTML page has some prerequisites concerning the browser workplace in which it is running. Therefore, it is per se not usable as a directly accessible page but needs to be embedded into a frame providing a defined set of functions.

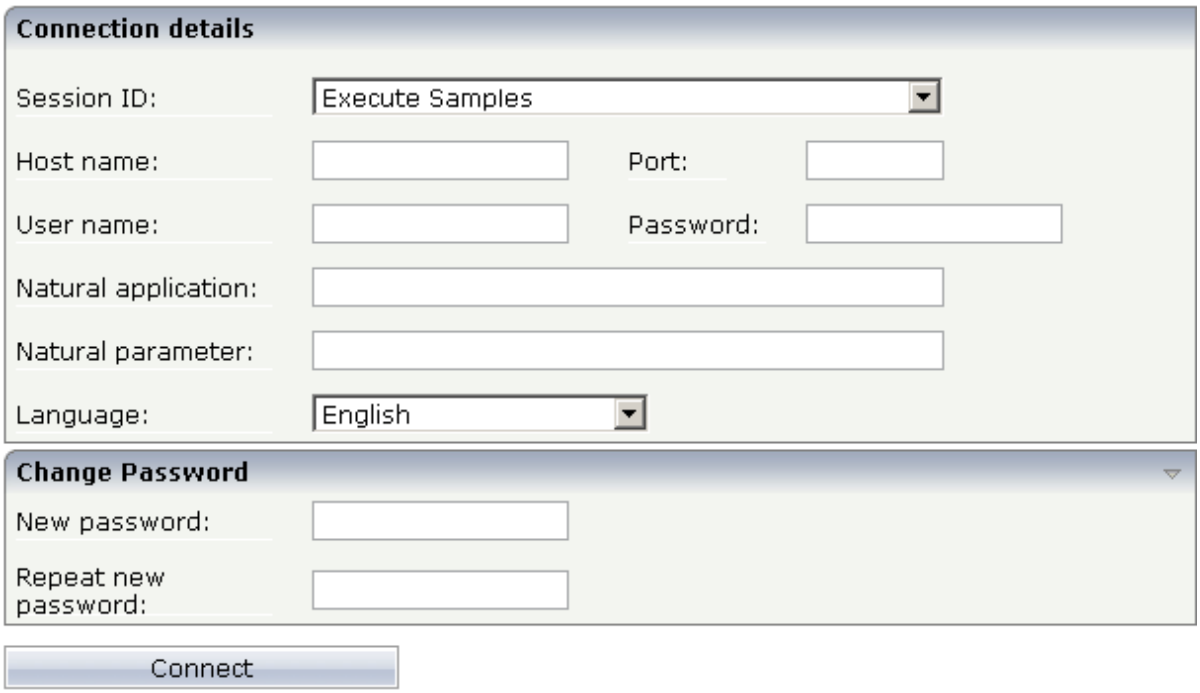
It is necessary to logon to Natural before starting an application. Therefore, Natural applications are started using a logon page.

To test the application

1. Enter the following URL inside your browser:

```
http://localhost:8080/cisnatural/servlet/StartCISPage?PAGEURL=/cisnatural/NatLogon.html
```

The logon page should now appear.



If the logon page is not displayed, check the following:

- URLs are case-sensitive. Double-check your input.
- Check whether the file *NatLogon.html* is available in the directory *cisnatural*.

2. On the logon page, select the entry **Execute samples** from the **Session ID** drop-down list box. You have prepared this entry earlier in this tutorial when you have set up the runtime environment.
3. Provide your user ID and password valid for the machine on which the Natural application will be running.
4. In the **Natural application** text box, enter the following information, depending on your Natural platform:

   - **Natural for Mainframes**
     Enter the name of the Natural program that is to be started. In our case, this is `HELLO-P`.
   - **Natural for UNIX**
     Enter the name of the UNIX shell script that is used to start Natural. By default, this is *nwo.sh*.
   - **Natural for Windows**
     Enter the name of the Windows command file (*.bat*) that is used to start Natural. By default, this is *nwo.bat*.

5. In the **Natural parameters** text box, enter the following information, depending on your Natural platform:

- **Natural for Mainframes**
  Enter the dynamic Natural profile parameters that are necessary to start your application:

  ```
  STACK=(LOGON CISHELLO)
  ```

  Note:
  With Natural for Mainframes, is recommended to specify the Natural program that starts the application in the **Natural application** text box instead of passing it with the profile parameter STACK.

- **Natural for UNIX and Natural for Windows**
  Enter the Natural command line that is necessary to start your application:

  ```
  STACK=(LOGON CISHELLO;HELLO-P)
  ```

6. Choose the **Connect** button.

   Your application should be started now.

7. Enter your name and choose the **Say Hello** button.

   The page should now successfully "talk" to your adapter.

You have now completed this tutorial. See the remaining section of these *First Steps* for some background information.