

Sessions For Character Interface Programs

Scott Augé

<https://www.linkedin.com/in/scottaugé/>

For those programmers who are working on shared character interface (aka green screen) programs with Progress ABL with little to no source code control, this is the program for you.

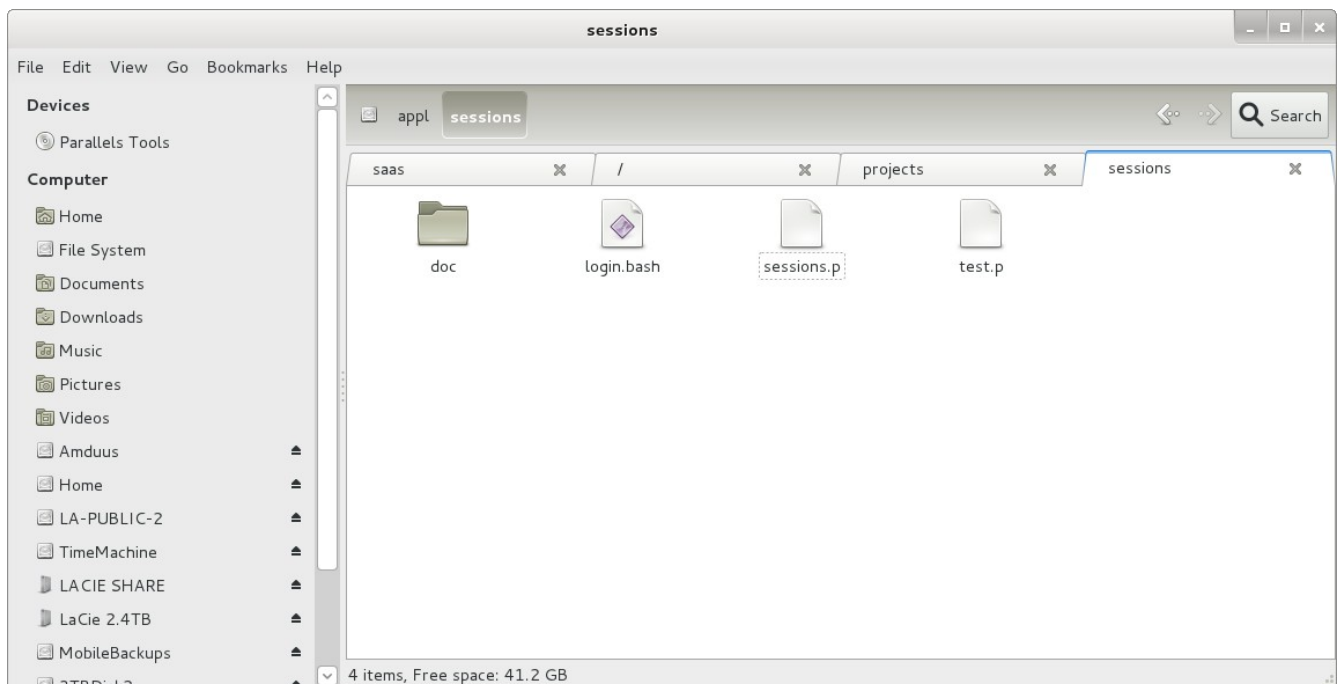
Originally the idea was for Webspeed applications to allow a transaction server to set different propaths based on the project number directories to work from, however I have encountered character development environments with pretty basic source code control.

Since this is “green screen” it is of course, oriented to a UNIX environment. If you update it to a Windows environment please let me know and I will create a new distribution package!

So what we are trying to achieve:

- The ability to create a set of directories that mimic the directory structure of the main application in the users work area named `$HOME/projects/projectnumber`;
- To divide those sets of directories according to a project number or string mnemonic (no spaces or special characters!).
- The ability to view the application using that project;
- Not to interrupt the work of other programmers until it becomes time to merge the code;
- To work on that code in `$HOME/projects/*` and keep the main branch free of locking source code.

First the file you will need to tailor to your environment:

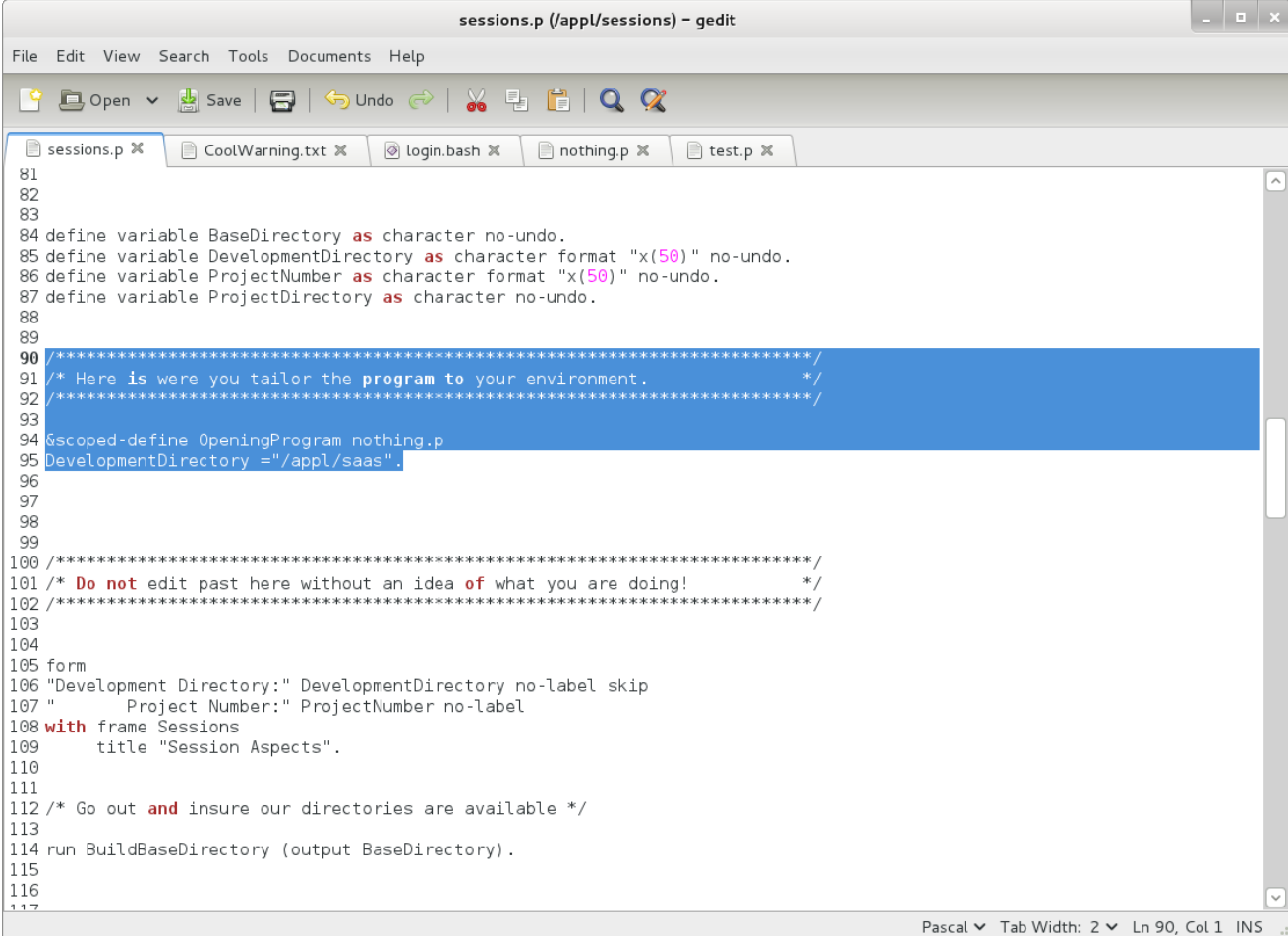


When you unpack the zip file you will find a directory structure much like the above (Linux GUI interface).

The `/doc` file contains the documentation for the session tool (aka this document).

The `test.p` and `login.bash` are files for testing if you wish to improve or play around with the tool. They should not be needed!

The `sessions.p` file is of interest to you and will need some minor tailoring:



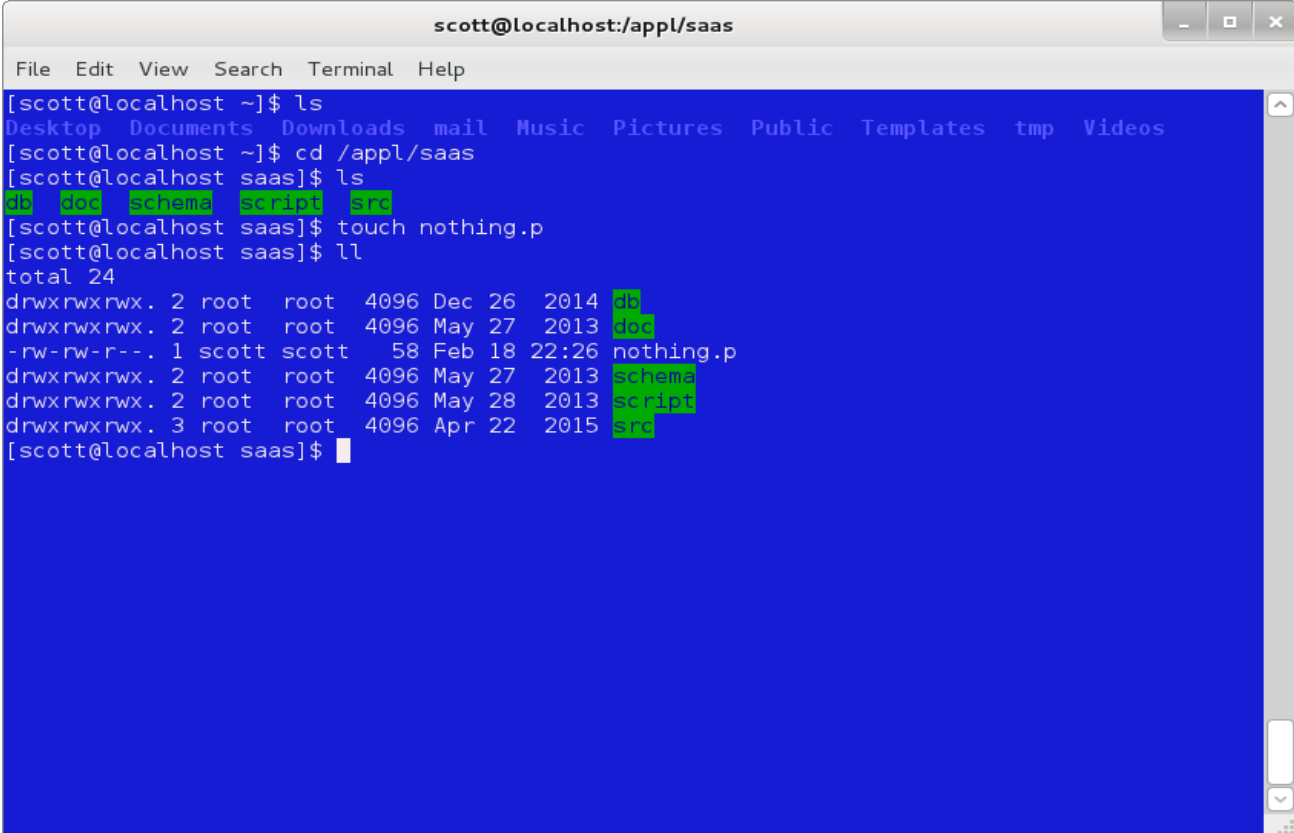
```
81
82
83
84 define variable BaseDirectory as character no-undo.
85 define variable DevelopmentDirectory as character format "x(50)" no-undo.
86 define variable ProjectNumber as character format "x(50)" no-undo.
87 define variable ProjectDirectory as character no-undo.
88
89
90 /*****
91 /* Here is were you tailor the program to your environment. */
92 /*****
93
94 &scoped-define OpeningProgram nothing.p
95 DevelopmentDirectory = "/appl/saas".
96
97
98
99
100 /*****
101 /* Do not edit past here without an idea of what you are doing! */
102 /*****
103
104
105 form
106 "Development Directory:" DevelopmentDirectory no-label skip
107 "      Project Number:" ProjectNumber no-label
108 with frame Sessions
109     title "Session Aspects".
110
111
112 /* Go out and insure our directories are available */
113
114 run BuildBaseDirectory (output BaseDirectory).
115
116
117
```

The `OpeningProgram` will be the entry point for your application – that is, the file on the command line following the `-p` parameter.

The next will be the `DevelopmentDirectory`. This should be the main directory for your source code – the `sessions.p` program uses it to mimic a directory structure like your code base. If you have more than one, you are going to have to decide which to use because this currently works with one directory.

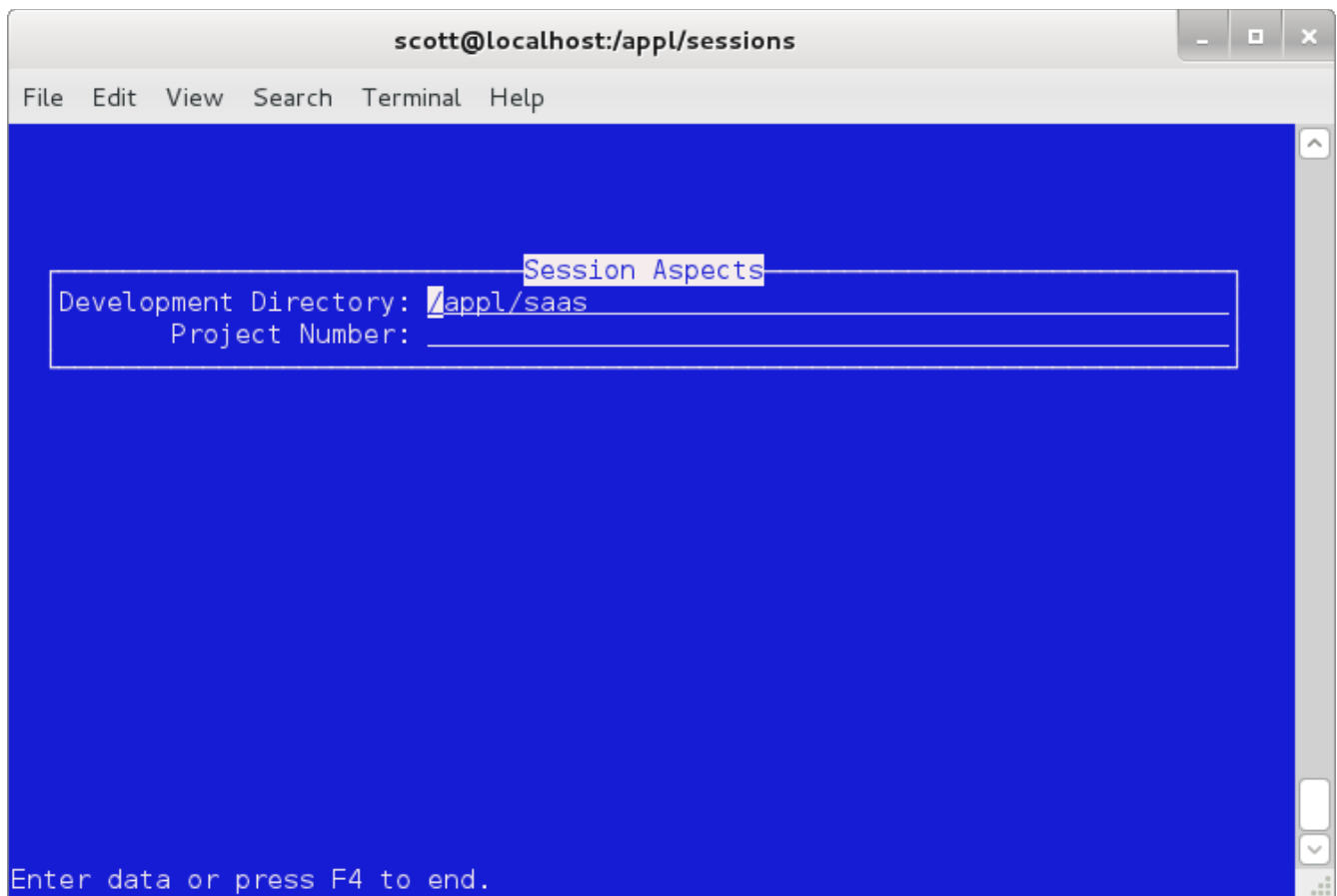
It does not copy code from that directory into your `$HOME/projects/projectnumber` directory. Just the directory structure (aka sub-directories.) You will be responsible for that!

Below is a sample coding application called Saas found in /appl/saas. I added a nothing.p because it is actually webspeed and has no character user interface. It was a good app to try this out on. :)

A terminal window titled "scott@localhost:/appl/saas" with a menu bar (File, Edit, View, Search, Terminal, Help) and standard window controls. The terminal output shows the following sequence of commands and results:

```
[scott@localhost ~]$ ls
Desktop  Documents  Downloads  mail  Music  Pictures  Public  Templates  tmp  Videos
[scott@localhost ~]$ cd /appl/saas
[scott@localhost saas]$ ls
db  doc  schema  script  src
[scott@localhost saas]$ touch nothing.p
[scott@localhost saas]$ ll
total 24
drwxrwxrwx. 2 root root 4096 Dec 26 2014 db
drwxrwxrwx. 2 root root 4096 May 27 2013 doc
-rw-rw-r--. 1 scott scott 58 Feb 18 22:26 nothing.p
drwxrwxrwx. 2 root root 4096 May 27 2013 schema
drwxrwxrwx. 2 root root 4096 May 28 2013 script
drwxrwxrwx. 3 root root 4096 Apr 22 2015 src
[scott@localhost saas]$
```

And your done! From the Progress editor run sessions.p and you will be presented the following screen:



For those with more than one branch in your PROPATH, the directory is provided for your input, but defaulted to DevelopmentDirectory. *Remember this is only from discovering the subdirectories of the directory entered.*

Following that is the Project Number (or name with no spaces as it is added to the \$HOME/projects directory in the propath.)

Enter that and if the project number does not exist, it will copy the directory structure from the DevelopmentDirectory into your \$HOME/projects/ProjectNumber directory. For example, if you had a project number 333147 a new directory \$HOME/projects/333147 would be added.

This new directory will then be added in the PROPATH before the current PROPATH so that those files show up for execution first.

Then the entry point program will be started and development can begin by copying files from the main branch into this project branch. (It is easily commented out.)

Remember to log out to the command line and then log back in so the PROPATH doesn't get stuffed up. Using it to work one project, the switching to another project, will leave the original projects directory as well the new projects directory in the PROPATH. Could be good, could be bad... only you know! (Code will be dependent on one project for the current project.)

Hints:

To get a list of which code was updated for the project, do the following:

```
cd $HOME/projects/yourproject  
find . -type f
```

That should make a listing of files that will need to go to main branch (hence no need to hunt and peck each directory if you have a huge application!)

Happy coding!