Perl How-To

Luca Bariani
LucaBariani@Ferrara.Linux.it

Version 1.0
December 2003

GNU Bayonne
http://www.gnu.org/software/bayonne

Ferrara Linux User Group
http://Ferrara.Linux.it

Contents

1 Introduction

GNU Bayonne can work with external scripting languages like Perl and Python. In this How-To we will show how using Bayonne with Perl (both 5.6.x and 5.8.x versions).

2 Before you begin

In this document we assume that:

- you have a Perl package installed;
- you are root in your local system (where Bayonne will be installed);
- you are familiar with Bayonne and its scripting language.

3 Perl version and path

Perl executable is usually located in `/usr/bin/perl` or in `/usr/local/bin/perl`, check your installation to understand your perl location (you can have different Perl versions in two different paths). To check which Perl is in your path use `which perl` (if it doesn't find anything perl executables is not in your path). To see your Perl version you need to execute `perl -v`, in this How-To we'll cover Perl versions 5.6.x and 5.8.x.

4 Bayonne installation

The packages commoncpp2, ccaudio and ccscript dont't use Perl, so you can install them as usual; running *configure* in the Bayonne source directory you must see a row like:

checking for perl... /usr/local/bin/perl
or
checking for perl... /usr/bin/perl

If you have different Perl versions from this row you can understand which version Bayonne has found (to change the Perl version to be used you can change your $PATH variable).
After this configuration you can run, as usual, *make* to compile and *make install* to install all. Note: in this How-To we'll use Bayonne 1.2.9.

5 Testing scripts

To test Bayonne and Perl we use the Bayonne script *testperl.scr*:

```
set %var1 8     # lowercase variable
set %var2 7     # lowercase variable
set %Var3 70    # uppercase variable
slog.info %var1 " " %var2 " " %Var3
libexec 20 scriptperl.pl %var1 %var2 %Var3  # tgi call
    # looking results
slog.info "Res: "%Res
slog.info "res2: "%res2
slog.info "Res3: "%Res3
slog.info "res4: "%res4
exit
```

and the Perl script *scriptperl.pl* (usually located in `/usr/local/libexec/bayonne`):

```
#!/usr/bin/perl
#or #!/usr/local/bin/perl
use lib '/usr/local/libexec/bayonne/'; # 1.2.x
use TGI;
$factor1 = $TGI::QUERY{'var1'}; # lowercase variable import
$factor2 = $TGI::QUERY{'var2'}; # lowercase variabel import
$v3 = $TGI::QUERY{'Var3'}; # uppercase variable import
print "imported variables: - $factor1 - $factor2 - $v3 -\n";
$result = $factor1*$factor2;  # any operation
TGI::set("Res",$result);  # uppercase variable export
TGI::set("res2",$result); # lowercase variable export
TGI::set("Res3",$v3); # uppercase variable export
TGI::set("res4",$v3); # lowercase variable export
exit
```

With these scripts we'll test variable import and export, both lowercase and uppercase.

6 Bayonne and Perl 5.6.x configuration

Bayonne has two different correct configurations for Perl 5.6.x (in this How-To we'll use Perl 5.6.1).

The first configuration is:

- in the *bayonne.conf* file you must have the row like `tgi=perl`;

(your Perl script don't need any executable right or any Perl binary path)

The second configuration is:

- in the *bayonne.conf* file you must have the row *tgi = xxx* **commented** (like `;tgi=perl`);
- the Perl script must have the correct Perl binary in the first row (like `#!/usr/bin/perl` or `#!/usr/local/bin/perl`); if you have different Perl executables in your system, be sure you are using the right one;
- the Perl script must be executable (`chmod +x scriptperl.pl`).

To test the last two points you can try executing the perl script from shell (like `/usr/local/libexec/bayonne/scriptperl.pl`).

At last you can run Bayonne (with `bayonne -x`) and execute testperl.scr script (with `bayonne --control start 0 testperl`).

If everything is correct your output is like:

```
fifo: cmd=start 0 testperl
dummy(0): testperl: 8 7 70
tgi: cmd=scriptperl.pl query=var1=8&var2=7&Var3=70 digits=
 clid=UNKNOWN dnid=UNKNOWN
fifo: cmd=wait 0 5687
tgi: exec /usr/local/libexec/bayonne/scriptperl.pl
fifo: cmd=imported variables: - 8 - 7 -  -
fifo: cmd=SET&0&Res&56
fifo: cmd=SET&0&res2&56
fifo: cmd=SET&0&Res3&
fifo: cmd=SET&0&res4&
fifo: cmd=exit 0 0
dummy(0): testperl: Res: 56
dummy(0): testperl: res2: 56
dummy(0): testperl: Res3:
dummy(0): testperl: res4:
```

From this example you can see that:

- lowercase variables are correctly imported from Bayonne to Perl;
- uppercase variables are not correctly imported from Bayonne to Perl;
- lowercase and uppercase variables are correctly imported from Perl to Bayonne.

7 Bayonne and Perl 5.8.x configuration

To configure Bayonne with Perl 5.8.x:

- in your *bayonne.conf* file you must have the row *tgi = xxx* **commented** (like `;tgi=perl`);
- your perl script must have the correct Perl binary in the first row (like `#!/usr/\-bin/perl` or `#!/usr/local/bin/perl`); if you have different Perl executables in your system, be sure you are using the right one;
- your perl script must be executable (`chmod +x scriptperl.pl`).

To test the last two points you can try executing the perl script from shell (like `/usr/local/libexec/bayonne/scriptperl.pl`).

At last you can run Bayonne (with `bayonne -x`) and execute testperl.scr script (with `bayonne --control start 0 testperl`).

If everything is correct your output is like:

```
fifo: cmd=start 0 testperl
dummy(0): testperl: 8 7 70
tgi: cmd=scriptperl.pl query=var1=8&var2=7&Var3=70 digits=
 clid=UNKNOWN dnid=UNKNOWN
fifo: cmd=wait 0 5687
tgi: exec /usr/local/libexec/bayonne/scriptperl.pl
fifo: cmd=imported variables: - 8 - 7 -  -
fifo: cmd=SET&0&Res&56
fifo: cmd=SET&0&res2&56
fifo: cmd=SET&0&Res3&
fifo: cmd=SET&0&res4&
fifo: cmd=exit 0 0
dummy(0): testperl: Res: 56
dummy(0): testperl: res2: 56
dummy(0): testperl: Res3:
dummy(0): testperl: res4:
```

From this example you can see that:

- lowercase variables are correctly imported from Bayonne to Perl;
- uppercase variables are not correctly imported from Bayonne to Perl;
- lowercase and uppercase variables are correctly imported from Perl to Bayonne.

8 Troubleshooting

If you use a wrong Bayonne/Perl configuration you can have errors or wrong executions:

- Perl 5.8.x with the row `tgi=perl` in the *bayonne.conf*: you can have an output like:

```
fifo: cmd=start 0 testperl
dummy(0): testperl: 8 7 70
tgi: cmd=scriptperl.pl query=var1=8&var2=7&Var3=70 digits=
             clid=UNKNOWN dnid=UNKNOWN
fifo: cmd=wait 0 2099
fifo: cmd=exit 0 0
dummy(0): testperl: Res:
dummy(0): testperl: res2:
dummy(0): testperl: Res3:
dummy(0): testperl: res4:
```

  In this case you don't see any explict error, but the Perl script is not executed.

- Perl 5.6.x (without the line `tgi=perl`) and Perl 5.8.x: using a Perl script without executable rights you can have an output like:

```
fifo: cmd=start 0 testperl
dummy(0): testperl: 8 7 70
tgi: cmd=scriptperl.pl query=var1=8&var2=7&Var3=70 digits=
             clid=UNKNOWN dnid=UNKNOWN
fifo: cmd=wait 0 2156
tgi: exec /usr/local/libexec/bayonne/scriptperl.pl
tgi: exec failed; /usr/local/libexec/bayonne/scriptperl.pl
fifo: cmd=exit 0 255
dummy(0): testperl: Res:
dummy(0): testperl: res2:
dummy(0): testperl: Res3:
dummy(0): testperl: res4:
```

  In this case you see an explicit error.

- Perl 5.6.x (without the line `tgi=perl`) and Perl 5.8.x: using a Perl script with a wrong perl executable path you can have an output like:

```
fifo: cmd=start 0 testperl
dummy(0): testperl: 8 7 70
tgi: cmd=scriptperl.pl query=var1=8&var2=7&Var3=70 digits=
             clid=UNKNOWN dnid=UNKNOWN
fifo: cmd=wait 0 2181
tgi: exec /usr/local/libexec/bayonne/scriptperl.pl
tgi: exec failed; /usr/local/libexec/bayonne/scriptperl.pl
fifo: cmd=exit 0 255
dummy(0): testperl: Res:
dummy(0): testperl: res2:
dummy(0): testperl: Res3:
dummy(0): testperl: res4:
```

In this case you see an explicit error.

9 Copyright