Extensions

Server-Side Web Languages

Uta Priss
School of Computing
Napier University, Edinburgh, UK

Outline

Libraries

Databases

Graphics

Libraries

For modern programming languages there are usually open-source archives on the web where users can contribute and download code libraries.

Before writing new code, it is advisable to check if it already exists and can be downloaded for free!

Caution

Free software can be of varying quality. This can be a security risk.

- Check the development status of the software: alpha, beta, production, released, mature
- Check the documentation and mailing lists
- Do a search on a search engine (Google) for the software and terms, such as “bugs”, “security”, etc.

Perl Modules

- www.cpan.org
  CPAN: Comprehensive Perl Archive Network,
- Sourceforge.net
  (Under “softwaremap”, Browse by programming language, Perl)

Using CPAN

Because some parts of modules can be written in C and need to be compiled, it can be difficult to install modules manually.

Fortunately, there is a command-line tool that makes things easy:

perl -MCPAN -e 'install MODULENAME'

Using Perl Modules

- Perl modules are scripts called "packagename.pm" (for example, CGI.pm).
- Modules are invoked with

  use packagename;

  at the beginning of a script.
- The path for modules is stored in the @INC array.
- Directories can be added to this path via the PERL5LIB environment variable or via

  use lib '/home/username/somedirectory';

PHP Extensions

- Sourceforge.net
  (Sourceforge is more important for PHP than for Perl)
- pear.php.net
  PHP Extension and Application Repository
- pecl.php.net
  PHP Extension Community Library (C extensions of PHP)

Databases

Server-side languages normally provide support for database connections.
Databases on the web are useful for

- Managing user data (logins and passwords)
- E-commerce, shopping carts
- Search engine data and other repositories

Database Requests

Database requests usually require the following sequence:

- connect to the database,
- prepare a query (as a string),
- execute the query,
- fetch the results (either row by row or as an array)
- finish the query (so that the database can clean up its buffers, this is optional)
- disconnect from the database

DBD versus DBI

Perl modules for databases normally consist of two parts:

- a database independent interface (DBI.pm)
- a database specific database driver (DBD.pm for generic use, Mysql.pm for Mysql; Pg.pm for Postgres, Oracle.pm for Oracle etc)

The advantage of this approach is that code can be written in a vendor-independent manner. The same code can be used for different DBMS.

Perl example

```
$db =
DBI->connect('dbi:mysql:yourdb;host=somehost','u','pwd');
$query = $db->prepare(qq{ SELECT * from table;});
$query->execute();
while (@row = $query->fetchrow()) {
     foreach $col (@row) {
          print "$col";
     }
     print '\n';
}
$db->disconnect;
```

Security

If form variables are directly inserted into database queries this can pose a security risk.

For example, a user could enter something like

*0; SELECT * from mysql.user; - -*

Any form input into queries needs to be carefully checked!

Graphics

If an application requires extensive use of interactive graphics, then using Java Applets or Java Servlets might be a better option than scripting languages.

But if the graphics are fairly static or can be generated on the fly as vector graphics, then scripting languages with graphics extensions might be suitable.

Vector Graphics

Vector graphics are line drawings, such as maps, UML diagrams, bar and pie charts.

On the WWW, vector graphics can be

- binary: Flash
- XML-based: SVG (Scalable Vector Graphics)
- HTML extensions: VRML

SVG: Scalable Vector Graphics

SVG is a great idea! Unfortunately, the industry has been slow to implement SVG and to agree on a common standard.
There are compatibility issues among different browsers.
A special viewer is required.

But the basic SVG functions are fairly compatible across browsers and not too difficult to implement.

The Perl SVG module

Example:

```
use SVG;
my $svg= SVG->new(width=>200,height=>200);
$svg->circle(id=>'1',cx=>100,cy=>100,r=>50,
           style =>{fill=>'red'});
print 'Content-type:  image/svg+xml\n\n';
my $out = $svg->xmlify(-dtd =>
'http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd')
print $out;
```