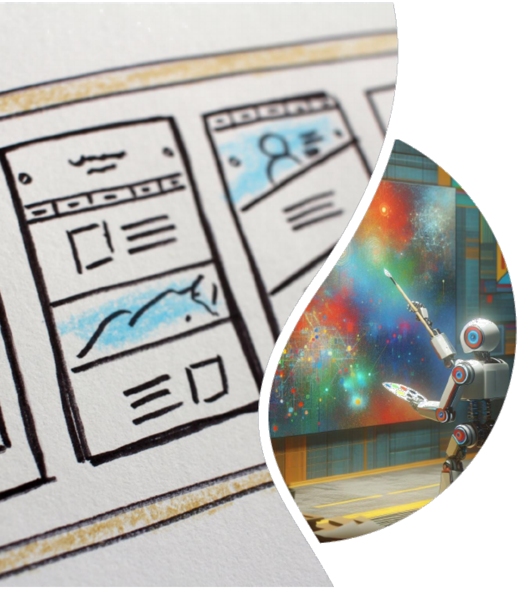

DGIN 5201 Digital Transformation Lecture 8

Lec 6: Back-end and Emails

Vlado Keselj

Time and date: 13:05–14:25, 23-Jan-2025
Location: LSC C236

Image: DALL-E. Bing Image Creator. Generated by AI

Previous Lecture

- Example e5: Backend server processing using CGI
  - CGI test: `test.cgi`
- Perl scripting language
- Preparing form for processing (`action="register.cgi"`)
- Return data display: `register.cgi`
- Example e6: Saving registration data
- Importance of file permissions in data protection

Example e7: Sending Registration by Email

- Use `rsync` to copy e6 to e7
- Modify the `register.cgi` file as follows by adding a new line:

```
...
&save_registration($name, $email, $certificate);
&send_email($name, $email, $certificate);
...
```

- and add the following subroutine at the end of the file:

```
sub send_email {
  my ($name, $email, $certificate) = @_;
  my $emailmessage = "To: vlado\@dnlp.ca\n".
    "Subject: New registration\n\n".
    "A new registration is received as follows:\n\n".
    "name: $name\nemail: $email\n".
    "certificate: $certificate\n";
  open(my $s, "|/usr/lib/sendmail -ti") or die;
  print $s $emailmessage;
  close($s);
}
```

Example e7: Sending Registration by Email (2)

- IMPORTANT: Instead of string `vlado@dnlp.ca` use your own email
- No not forget to use backslash (\) just before the at-sign (@) in email, as in `vlado\@dnlp.ca` because the string is delimited by double-quotes. Otherwise, Perl will replace `@dnlp` with the value of that array
- Test the program and make sure that you receive email after each registration

Example e7: Received Email

- If everything is implemented correctly, and if it works, you should receive an email similar to:

```
From: "...your name..." <YourCSID@willow.cs.dal.ca>
Date: Thu, 23 Jan 2025 13:30:34 -0400 (AST)
To: your_email@dal.ca
Subject: New registration

A new registration is received as follows:

name: Test Name
email: test-email@cs.dal.ca
certificate: DB
```

Example e8: Testing Other Scripting Languages

- Copy e7 to e8 using rsync
- Update `.htaccess` to use passwords from `e8/.htpasswd`
- Create files index-php.html and index-py.html to use PHP and Python as actions: register.php and register.py
- Implement basic register.php and register-py.cgi to print filled form

Example e8: Testing a PHP Script: register.php

```
<html><head><title>Applicant Registration</title></head>
<body>
<h1>Registration</h1>

<p>The following registration is received:

<table>
<tr><th align=right>First and last name:</th>
<td><?php echo $_POST['name'] ?></td></tr>
<tr><th align=right>Email:</th>
<td><?php echo $_POST['email'] ?></td></tr>
<tr><th align=right>Certificate (DB, HI, DS):</th>
<td><?php echo $_POST['certificate'] ?></td></tr>
<tr><td align=center colspan=2>
<a href="index-php.html">Back to Registration Page</a>
</td></tr></table>
```

Example e8: Testing a Python Script: register-py.cgi

```
#!/usr/bin/python
import cgi
print "Content-type: text/html\n\n"
print "<html><body><h1>Registration</h1>\n";
print "<p>The following registration is received:\n";

form=cgi.FieldStorage()
name  = form.getvalue('name')
email = form.getvalue('email')
certificate  = form.getvalue('certificate')
print """<table><tr><th align=right>First and last name:</th>
<td>"""+name+"""</td></tr>
<tr><th align=right>Email:</th><td>"""+email+"""</td></tr>
<tr><th>Certificate (DB, HI, DS):</th>
<td>"""+certificate+"""</td></tr>
<tr><td align=center colspan=2>
<a href="index-py.html">Back to Registration Page</a></td>
</tr></table>\n"""
```

Example e8: Renaming Python Script to `register.py`

- We can copy `register-py.cgi` to `register.py` and try if it works (use `index-py2.html` as the index page)
- It does not!   (i.e.., probably does not)
- Solution: Add the following line to `.htaccess` file:

```
AddHandler cgi-script .py
```

Scripting Languages

- Developed as helpful tools for automating tasks, rapid prototyping, gluing together other programs
- Evolved into mainstream programming tools
- Examples
  - shell scripts (e.g., bash)
  - Early text processing: sed, Awk
  - Perl, PHP, Python, Ruby, Tcl, Lua, . . .
  - Javascript
  - Visual Basic, VBScript, JScript, CScript, WScript,. . .
  - . . .

Brief Overview of some Programming Languages

- (by Brian Kernighan)
- 1940's — machine language
- 1950's — assembly language
- 1960's — high-level languages: Fortran, Algol, Cobol, Basic
- 1970's — systems programming: C, but also Pascal
- 1980's — object-oriented: Smalltalk, C++
- 1990's — strongly-hyped: Java, modest beginning of JavaScript
- 2000's — lookalike languages: C#, PHP
- 2010's — retry? Scala, Go, Rust, Swift

Overview of Programming (Scripting) Languages

- 1940's — (machine language)
- 1950's — (assembly language)
- 1960's — Fortran, Algol, Cobol — Basic, Snobol
- 1970's — systems programming: C, Pascal — shell
- 1980's — OOP: Smalltalk, C++ — awk, Perl
- 1990's — Web: Java — Perl, Python, PHP
- 2000's — Frameworks: C# — JavaScript
- 2010's — retry? Scala, Go, Rust, Swift — Typescript

Typical Characteristics of Scripting Languages

- Interpreted
- Garbage collection
- Weakly typed; minimal use of types and declarations
- Text strings as an important data type
- Regular expressions support
- Easy execution of external programs